CSS3 デザインブック

仕事で絶対に使うプロのテクニック

web creators

よく見るあの表現から
最先端のテクニックまで

サンプルデータをダウンロードできます!!

CSS3でできること・したいこと ぜんぶ 1冊でわかります!!

CSS／CSS3の基本も解説！

［IE対策を踏まえた実践テクニック］
［わかりやすい解説で応用力もUP!］
［スマホサイトのアイデアも充実！］

CHAPTER 01

レイアウト

- レスポンシブWebデザイン
- flexboxを使ったボックスレイアウト
- ナビゲーション固定のWebページ
- リキッドタイプのフリーレイアウト
- レスポンシブなグリッドレイアウト
- 透過画像を組み合わせたデザイン
- アナログテイストの表現
- テキストを読ませるレイアウト

CHAPTER 02

ユーザーインターフェイス

- CSSで実装するプルダウン機能
- タブ型ナビゲーション
- 画像を使わないパンくず型フロー
- 問い合わせフォーム
- 縦型プルダウンメニュー
- ロールオーバーを使った演出
- スマートフォンのUIフォーム
- スライドするリスト型ナビゲーション
- 文字数の増減に対応したリスト

CHAPTER 03

グラフィック・タイポグラフィ

- インタラクティブなフォトギャラリー
- border-imageを使った写真フレーム
- アニメーションを使ったマスク
- Webフォントでテキストを表現
- 属性セレクタを利用した写真の装飾
- text-shadowでつくる立体的な文字
- 背景画像の多彩な表現
- DropCapsを使った雑誌風の文字組み
- ドロップシャドウ効果

CHAPTER 04

デザインパーツ

- グラデーションのボタンデザイン
- HSLA指定を利用したアニメーション
- 可読性の高いテーブル
- マークアップだけでつくるツリー図
- outlineプロパティを使った飾り罫
- 角丸の見出し・タブメニュー
- 属性セレクタを使ったリスト表現
- スマートフォン向けデザインパーツ
- 蛍光マーカー風の装飾

www.MdN.co.jp
MdN
エムディエヌコーポレーション

CSS3 デザインブック

仕事で絶対に使うプロのテクニック

web creators

www.MdN.co.jp
MdN
エムディエヌコーポレーション

CSS3
デザインブック

仕事で絶対に使うプロのテクニック

web creators

はじめに

CSS（Cascading Style Sheets）は、HTML ／ XHTMLで構造化された文書情報に対して、ページ全体のレイアウト、文字（フォント）のサイズ・色、個々の要素の配置や装飾といった“見た目”を指定する技術です。その最新バージョンであり、現在策定中のCSS3では新しくプロパティやセレクタが追加され、例えば属性セレクタを利用してリンクファイルの拡張子別にアイコン画像を表示する指定が可能になったり、文字に影をつける、ボタンの角を丸くするなどの装飾が画像を用いずに表現できるようになったりと、CSS3によってWebサイトの表現の幅は格段に広がります。その一方で、Internet Explorer 8以下のバージョンのブラウザはCSS3に対応していないことから、Web制作の実務案件でCSS3を使うことをためらう方も多いのではないでしょうか。

本書は、Web制作の現場で「CSS3をどのように使うか」を、サンプルを例にソースコードとともに解説したテクニック集です。各記事で扱っているテクニックやサンプルには、実際にWebサイトでよく使用されている実用的なものを選び、Web制作の第一線で活躍しているプロが制作手法を丁寧に解説しています。CSS3を実務案件で使うことを前提に、対応していないブラウザへの対策やアクセシビリティを確保する手法なども、できる限り盛り込んだ実践的な内容になっています。また、コピー＆ペーストにとどまらないコーディングの応用力も身に付けられるように、単純なソースコードの説明だけに終始せず「なぜこう書くのか」という理論的な背景もフォローしました。

CSS3はまだ策定中の技術ではあるものの、段階的にブラウザへの実装が進められており、スマートフォンサイトでは実質的なスタンダードとなっています。仕様の確定にはまだ時間が掛かるでしょうが、使える技術を部分的にでも取り入れていくことでコーディングがシンプルになりますし、将来の移行もよりスムーズに行えるはずです。本書が一人でも多くの方のお役に立ち、CSS3を活用することで、より使いやすく華やかなWebサイトやコンテンツが生み出される手助けになれば幸いです。

MdN編集部

CSS3デザインブック
仕事で絶対に使うプロのテクニック

CONTEN

INTRODUCTION

CSS3の基本解説

CHAPTER 01

レイアウト

TS

CHAPTER 02

ユーザー
インターフェイス

CONTEN

CHAPTER 03

グラフィック・タイポグラフィ

CHAPTER 04

デザインパーツ

TS

本書の使い方

本書はCSS3を利用したWebページの制作手法をステップ・バイ・ステップ形式で解説したテクニック集です。
本書は次のような構成になっています。

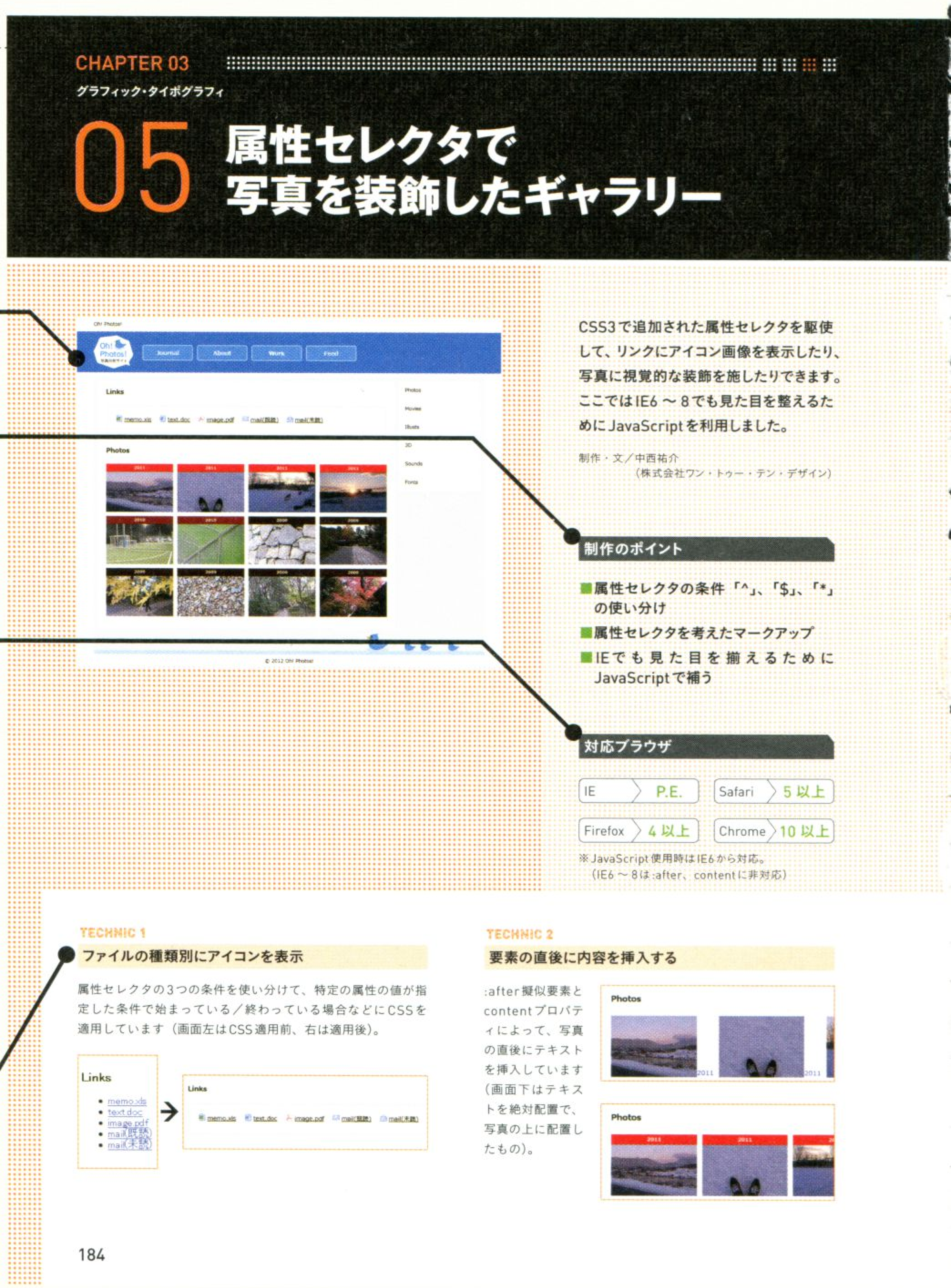
CHAPTER 01　レイアウト

Webページ全体のレイアウトに関わるテクニックを扱っています。2カラム、3カラムなどの段組みを利用したレイアウトやグリッドに沿ったレイアウトなど、Webページの大枠をつくる手法です。

CHAPTER 02　ユーザーインターフェイス

ナビゲーションメニューやユーザーフォームなど、Webページのインターフェイスに関するテクニックを紹介しています。インターフェイスの見やすさ、使いやすさは、Webページの印象を左右する大切なポイントです。

紙面の構成


CHAPTER 03
グラフィック・タイポグラフィ
05 属性セレクタで写真を装飾したギャラリー

CSS3で追加された属性セレクタを駆使して、リンクにアイコン画像を表示したり、写真に視覚的な装飾を施したりできます。ここではIE6～8でも見た目を整えるためにJavaScriptを利用しました。

制作・文／中西祐介
（株式会社ワン・トゥー・テン・デザイン）

制作のポイント

■属性セレクタの条件「^」、「$」、「*」の使い分け
■属性セレクタを考えたマークアップ
■IEでも見た目を揃えるためにJavaScriptで補う

対応ブラウザ

IE P.E.　Safari 5以上
Firefox 4以上　Chrome 10以上

※JavaScript使用時はIE6から対応。
（IE6～8は:after、contentに非対応）

TECHNIC 1
ファイルの種類別にアイコンを表示

属性セレクタの3つの条件を使い分けて、特定の属性の値が指定した条件で始まっている／終わっている場合などにCSSを適用しています（画面左はCSS適用前、右は適用後）。

TECHNIC 2
要素の直後に内容を挿入する

:after擬似要素とcontentプロパティによって、写真の直後にテキストを挿入しています（画面下はテキストを絶対配置で、写真の上に配置したもの）。

184

サンプル

本編で制作しているサンプルの見本です。

制作のポイント

解説を理解する上で、押さえておきたいコーディング上の重要なポイントを紹介しています。

対応ブラウザ

サンプルが対応しているブラウザ、およびバージョンの情報を示しています。おもにInternet Explorerは6以上、Safari、Firefox、Chromeについては原稿執筆時点の最新版を対象にしています。サブバージョンは必要に応じて記載しております。
また「P.E.」と記載しているものは、プログレッシブ・エンハンスメントを採用して対応させているものです（プログレッシブ・エンハンスメントについてはINTRODUCTIONの16ページの解説もご覧ください）。
スマートフォンサイトの記事では、iPhone（iOS）、Androidでの標準ブラウザでの検証結果を記載しています。記載しているバージョンはOSのバージョンです。

TECHNIC

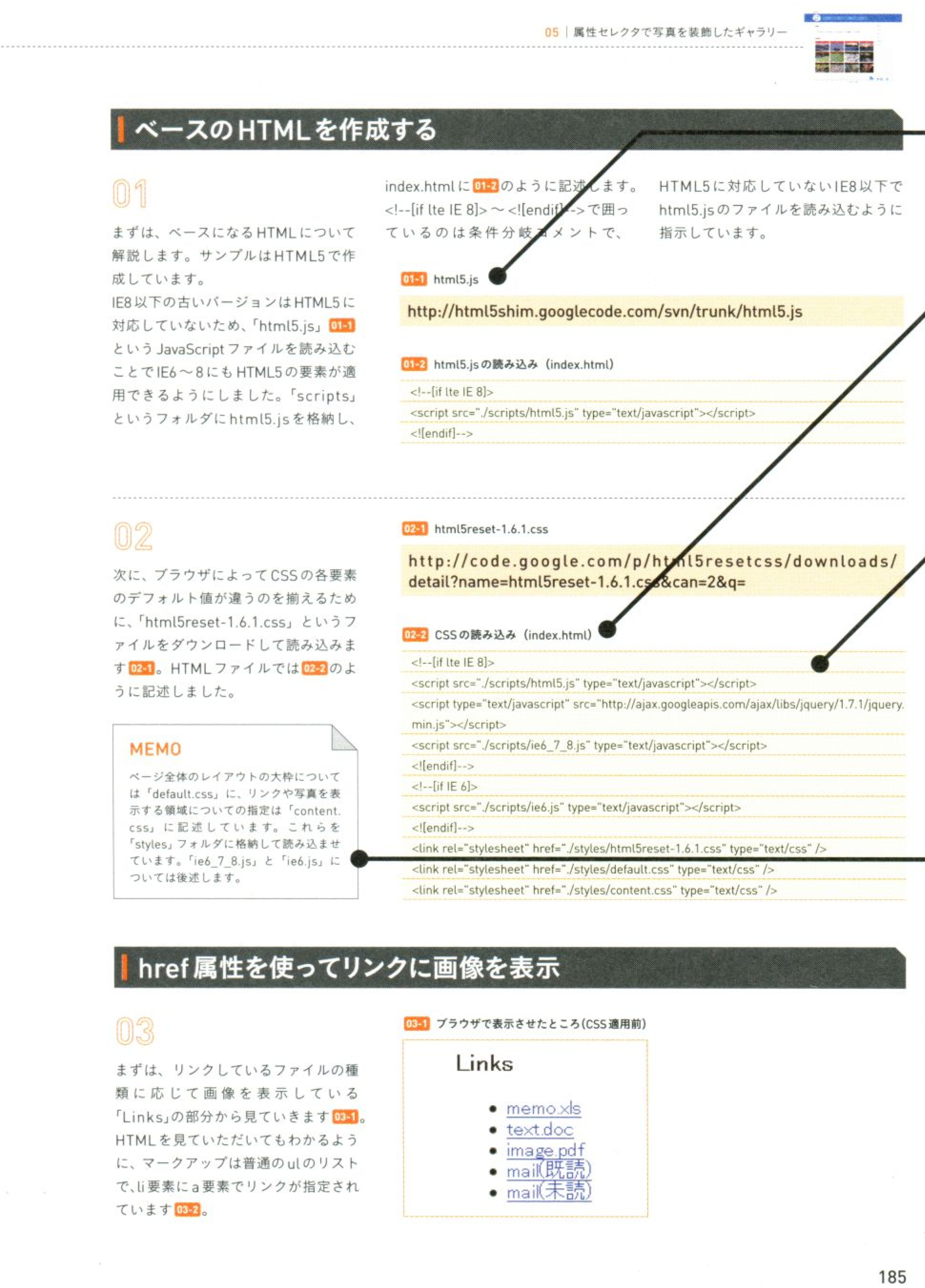
サンプルを制作する上で、どのようなテクニックやデザイン的なアイデアが盛り込まれているかを紹介しています。

CHAPTER 03　グラフィック・タイポグラフィ

写真画像の装飾や文字の表現に関するテクニックを紹介しています。ユーザーが興味を引かれる、視覚的に美しい・心地よいWebページをつくるには、画像や文字を装飾する手法が欠かせない要素になります。

CHAPTER 04　デザインパーツ

ボタン、罫線、背景など、デザインパーツの装飾方法を紹介しています。CSS3を利用することで、これまでは画像を使うことでしか実現できなかった装飾がスタイル指定で可能になるなど、表現の幅が広がります。



05 | 属性セレクタで写真を装飾したギャラリー

ベースのHTMLを作成する

01

まずは、ベースになるHTMLについて解説します。サンプルはHTML5で作成しています。IE8以下の古いバージョンはHTML5に対応していないため、「html5.js」01-1 という JavaScript ファイルを読み込むことでIE6～8にもHTML5の要素が適用できるようにしました。「scripts」というフォルダにhtml5.jsを格納し、index.htmlに01-2のように記述します。<!--[if lte IE 8]>～<![endif]-->で囲っているのは条件分岐コメントで、HTML5に対応していないIE8以下でhtml5.jsのファイルを読み込むように指示しています。

01-1 html5.js

http://html5shim.googlecode.com/svn/trunk/html5.js

01-2 html5.jsの読み込み（index.html）

```
<!--[if lte IE 8]>
<script src="./scripts/html5.js" type="text/javascript"></script>
<![endif]-->
```

02

次に、ブラウザによってCSSの各要素のデフォルト値が違うのを揃えるために、「html5reset-1.6.1.css」というファイルをダウンロードして読み込みます 02-1。HTMLファイルでは 02-2 のように記述しました。

MEMO

ページ全体のレイアウトの大枠については「default.css」に、リンクや写真を表示する領域についての指定は「content.css」に記述しています。これらを「styles」フォルダに格納して読み込ませています。「ie6_7_8.js」と「ie6.js」については後述します。

02-1 html5reset-1.6.1.css

http://code.google.com/p/html5resetcss/downloads/detail?name=html5reset-1.6.1.css&can=2&q=

02-2 CSSの読み込み（index.html）

```
<!--[if lte IE 8]>
<script src="./scripts/html5.js" type="text/javascript"></script>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
<script src="./scripts/ie6_7_8.js" type="text/javascript"></script>
<![endif]-->
<!--[if IE 6]>
<script src="./scripts/ie6.js" type="text/javascript"></script>
<![endif]-->
<link rel="stylesheet" href="./styles/html5reset-1.6.1.css" type="text/css" />
<link rel="stylesheet" href="./styles/default.css" type="text/css" />
<link rel="stylesheet" href="./styles/content.css" type="text/css" />
```

href属性を使ってリンクに画像を表示

03

まずは、リンクしているファイルの種類に応じて画像を表示している「Links」の部分から見ていきます03-1。HTMLを見ていただいてもわかるように、マークアップは普通のulのリストで、li要素にa要素でリンクが指定されています03-2。

03-1 ブラウザで表示させたところ(CSS適用前)

Links

- memo.xls
- text.doc
- image.pdf
- mail(既読)
- mail(未読)

185

使用する外部ファイル

サンプルで使用しているプラグインやライブラリなどを、URLを併記して記載しています。

コードの情報

サンプルのファイル群のなかで、該当コードが記述されているファイル名、およびコードの内容等について記載しています。

コード

ステップのなかで記述しているコードです。改行されていても行間にラインがない場合は、紙面の都合上で改行されているものの、実際のコードは1行で記述されていることを示しています。解説のために行番号が付されている場合もあります。

MEMO

TIPS

ATTENTION

COLUMN

解説文中で触れられなかった補足情報、仕事で応用する際に知っておきたいポイントや注意点などを記載しています。

※本書に掲載されているURL、サイト名、CSS、およびライブラリなどに関する情報は2012年1月現在のものです。

本書のサンプルデータについて

サンプルデータのダウンロード
本書に掲載されているサンプルデータは、弊社Webサイトの本書のページよりダウンロードできます。
URLは下記のとおりです。

http://www.MdN.co.jp/di/book/6253/

- ダウンロードしたファイルはZIP形式で保存されています。
- Windows、Macintoshそれぞれの解凍ソフトを使って圧縮ファイルを解凍してください。
- ZIPファイルはCHAPTERごとに分かれています。
- 解凍したフォルダ内には各記事番号のフォルダが収録されています。
- 各記事番号のフォルダには、本文で解説しているHTMLファイルやCSSファイル、JSファイル、画像等が収録されています。
- サンプルファイルには「はじめにお読みください.html」ファイルが同梱されていますので、ご使用の前に必ず先にお読みください。

サンプルデータに関するご注意

- 弊社Webサイトからダウンロードできるサンプルデータは、本書の解説内容をご理解いただくために、ご自身で試される場合にのみ使用できる参照用データです。その他の用途での使用や配布などは一切できませんので、あらかじめご了承ください。ただし、サンプルデータ中に含まれる一般に公開されているライブラリ類については、各ライブラリのライセンス条件に従います。
- 弊社Webサイトからダウンロードできるサンプルデータの著作権は、それぞれの制作者に帰属します。
- 弊社Webサイトからダウンロードできるサンプルデータを実行した結果については、著者および株式会社エムディエヌコーポレーションは一切の責任を負いかねます。お客様の責任においてご利用ください。
- 本書に掲載されているHTML、CSS、JavaScript等のコメントや改行位置は、紙面掲載用として加工していることがあります。ダウンロードしたサンプルデータとは異なる場合がありますので、あらかじめご了承ください。
- サンプルのなかには、Webサーバーの機能を利用するなどの理由から、ローカル環境上では正しく動作しないものがあります。このようなサンプルの場合は、Webサーバー上に設置したうえで機能をお試しください。

INTRODUCTION

CSS3の基本解説

CSS3 Basic Lesson

解説／境 祐司

CSS3を理解する

CSS（Cascading Style Sheets）は、HTMLおよびXHTMLでマークアップされた構造化文書の視覚表現を指定するための技術です。CSS3は、現在策定中の新しい仕様ですが、基本文法などは変わりません。まずは、CSSの基礎について理解しておきましょう。

CSSの役割と使い方

印刷媒体とWebページの違い

Webページの特徴は、構造化するための技術（HTML）とプレゼンテーション技術（CSS）によって形作られていることです 01。印刷媒体と比較してみましょう。

紙に印刷された書籍や雑誌は、人間にとって見やすい・読みやすい紙面で提供され、情感に訴える魅力的なビジュアルデザインになっています。また、テキストが中心の媒体などは、可読性や視認性が重視されていることは言うまでもありません。

Webページもブラウザ上では紙媒体と同様ですが、人間だけではなく、機械（プログラム）に対しても正しい情報を伝える必要があります。なぜなら、印刷物とは異なり、さまざまな機器で閲覧されるからです。ブラウザが搭載されている機器は、パソコンだけではなく、携帯電話からスマートフォン、タブレット、スマートテレビなど、多種多様です。1種類のデザインでは対応できないことがわかると思います 02。

HTMLとCSSは常にセットで考える

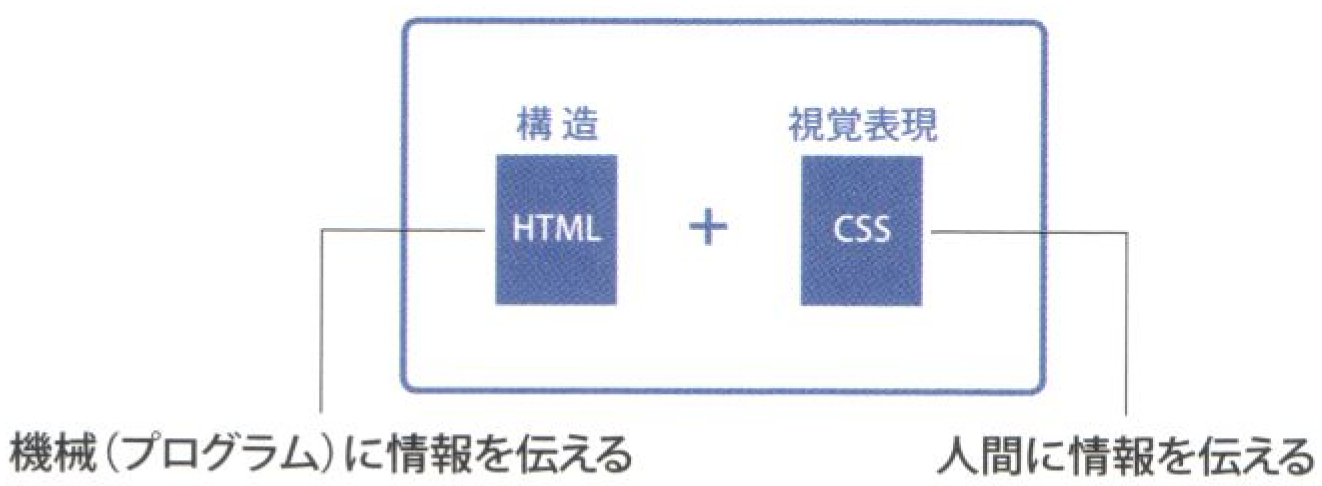
機械（プログラム）が理解できるように記述された文書を「マシン・リーダブル」と表現します。人間は、文字の大きさや太さ、配色、配置などを見て、ページの見出しなのか、記事の小見出しなのか判断することができますが、機械は文書の意味を正しく読み取ることができません。

そこで、見出しや段落、図版などに専用のタグ（HTMLの要素や属性）を付けて、意味を定義します。この作業を「マークアップ」と呼びます。

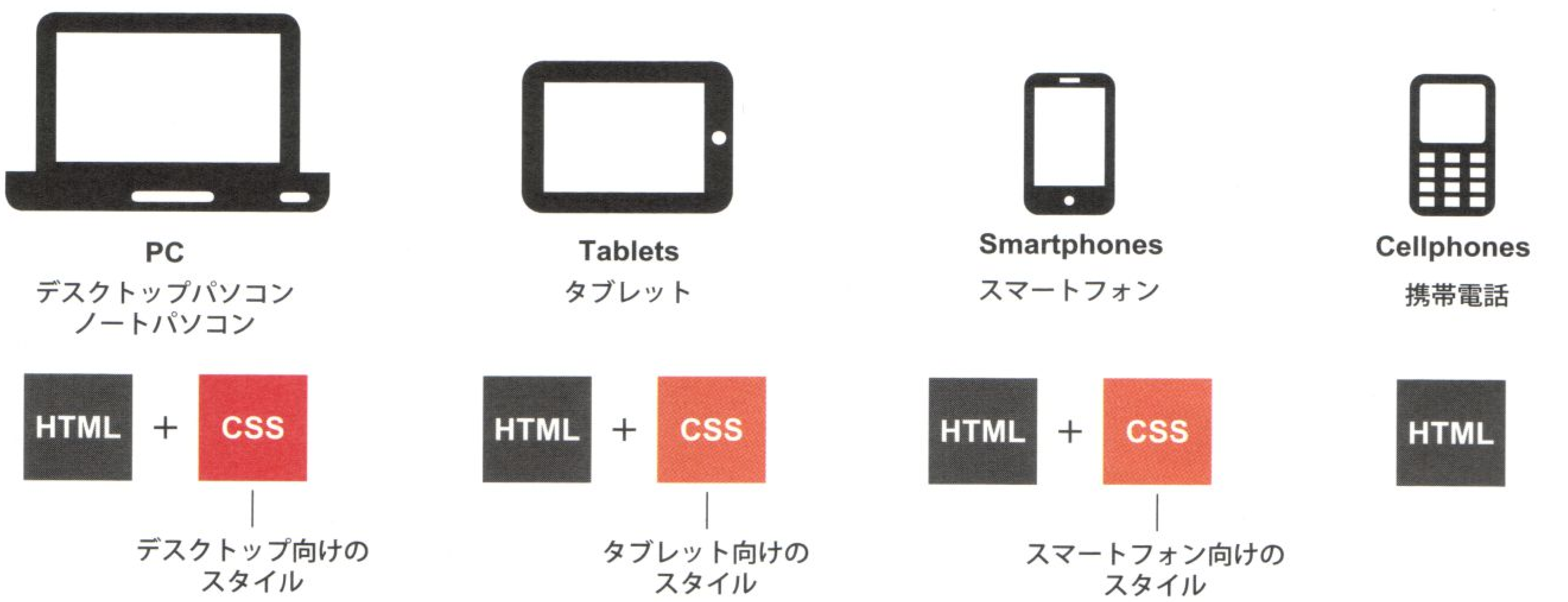
マークアップされた文書は、ブラウザ上には表示されず、あくまで機械に意味を伝えるための情報として扱われます。つまり、HTMLは文書の意味を定義するためのマークアップ言語であり、ブラウザ上の見栄えを決めるときはCSSを使用しなければいけないのです。Webページの可読性やビジュアルの美しさなど、すべての視覚表現はCSSによって作られます。

高性能パソコンでも、非力な携帯電話でも、マシンリーダブルなHTMLファイルであれば、文書の意味を解釈することができます。また、見出しだけを表示したり、図版を非表示にするなど、機器の制限に合わせて最適化することができます。パソコンのディスプレイなど、大きなスクリ

01 HTMLとCSSの役割



02 Webページを閲覧する多様な機器



ーンでは（段組みレイアウトなど）印刷物に近い表現が可能になりますが、携帯電話やスマートフォンなどの小さなスクリーンでは、可能な限りシンプルに表示したほうが見やすくなります。また、弱視の方や高齢者に対しては文字を大きく表示したり、視覚障がい者向けに音声読み上げで情報を提供することも可能になります。

さまざまな機器で閲覧されるWebページでは、構造と見栄えの分離がいかに重要か理解できたと思います。見栄えのために、機械が解釈しにくい構造にならないように注意しましょう（見出しを画像でグラフィカルに表現する際、代替テキストを記述しないなど）。CSSは、人間のための視覚表現技術ですが、常にマークアップ言語（HTML）とセットで考えなくてはいけません。

CSS3のモジュールについて

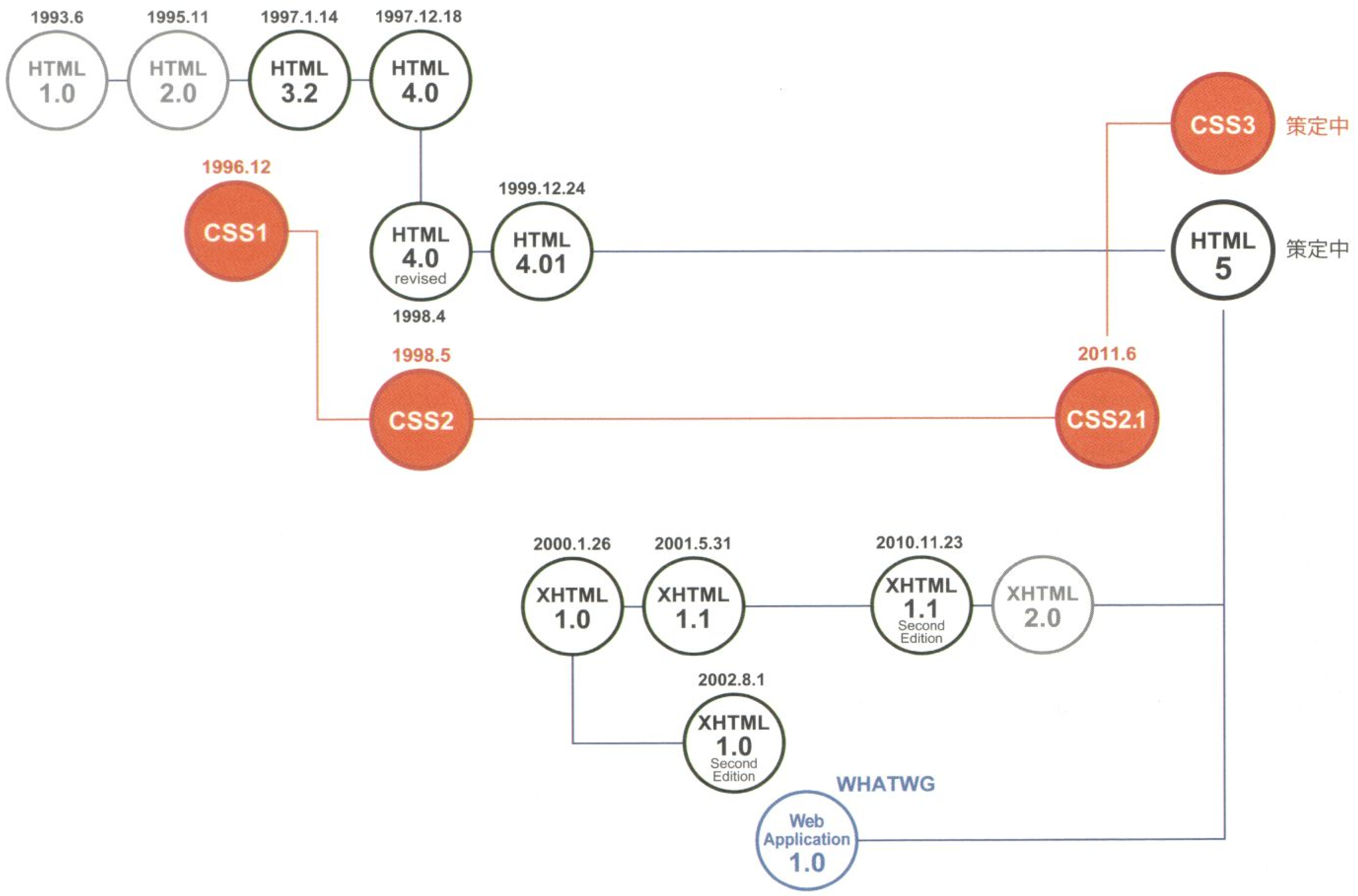
CSS（Cascading Style Sheets：カスケーディング・スタイルシート）は、Web技術の標準化を推進している組織「W3C」によって策定されています。

CSSの最初のバージョンは、1996年12月に勧告されました。W3Cは、草案、最終草案、勧告候補、勧告案、勧告という長いプロセスを経て、技術仕様をリリースしています。勧告には、強制力はありませんが、国際的な標準規格として受け入れられています。現在普及しているCSS2は1998年5月に勧告、2011年6月には改訂版となるCSS2.1が勧告されました 03。

現在策定中のCSS3は、膨大かつ複雑な仕様になっているため、機能ごとに「モジュール」として管理され、開発が進められています。例えば、色に関する「CSS Color Module Level 3」は勧告案、段組みレイアウトを指定する「CSS Multi-column Layout Module」は勧告候補、縦書きなどの仕様が含まれる「CSS Writing Modes」は草案、という状況です（2012年1月現在）。

主要なブラウザは、これらのモジュールの一部を先行実装していますが、サポートしている仕様は異なっています。アップデートによって実装状況が変化していきますので、CSS3を使用する場合は、常に最新の情報を収集しておく必要があります。W3Cのサイト 04 05 には、モジュールごとの開発状況を一覧することができる「CSS current work & how to participate」ページが用意されていますので、定期的にチェックしておきましょう。

03 W3Cが策定するHTMLとCSSの歴史



04 W3Cのサイトで公開されているCSSの総合ページ

http://www.w3.org/Style/CSS/

05 CSS3のモジュールごとの開発状況を一覧できるページ

http://www.w3.org/Style/CSS/current-work

CSSの基本文法（記述方法）

CSS基本文法の習得は、「セレクタ」、「プロパティ」、「プロパティの値」を理解することから始まります。プログラミングのような高度な専門性は必要ありませんので、短期間でマスターできるでしょう。

基本的な記述ルール

CSSの基本的な記述ルールは、「セレクタ { プロパティ : 値 }」となります 06。

例えば、ページの大見出しの色を赤で表現したい場合は、「ページの大見出し { 文字の色 : 赤 }」ですから、CSSでは「h1 { color : red }」と記述します。プロパティと値の間はコロン「:」、複数の指定が続く場合は、行末ごとにセミコロン「;」を付けなくてはいけません。「h1 { color : red ; font-size : 1em }」のようになります。

h1 〜 h6やpなど、HTMLの要素名を対象としたセレクタを「タイプセレクタ」と呼びます。ページ全体にスタイルを指定したいのであれば、body要素を対象にして「body { color: red }」のように記述することができます 07 08 09 10 11。

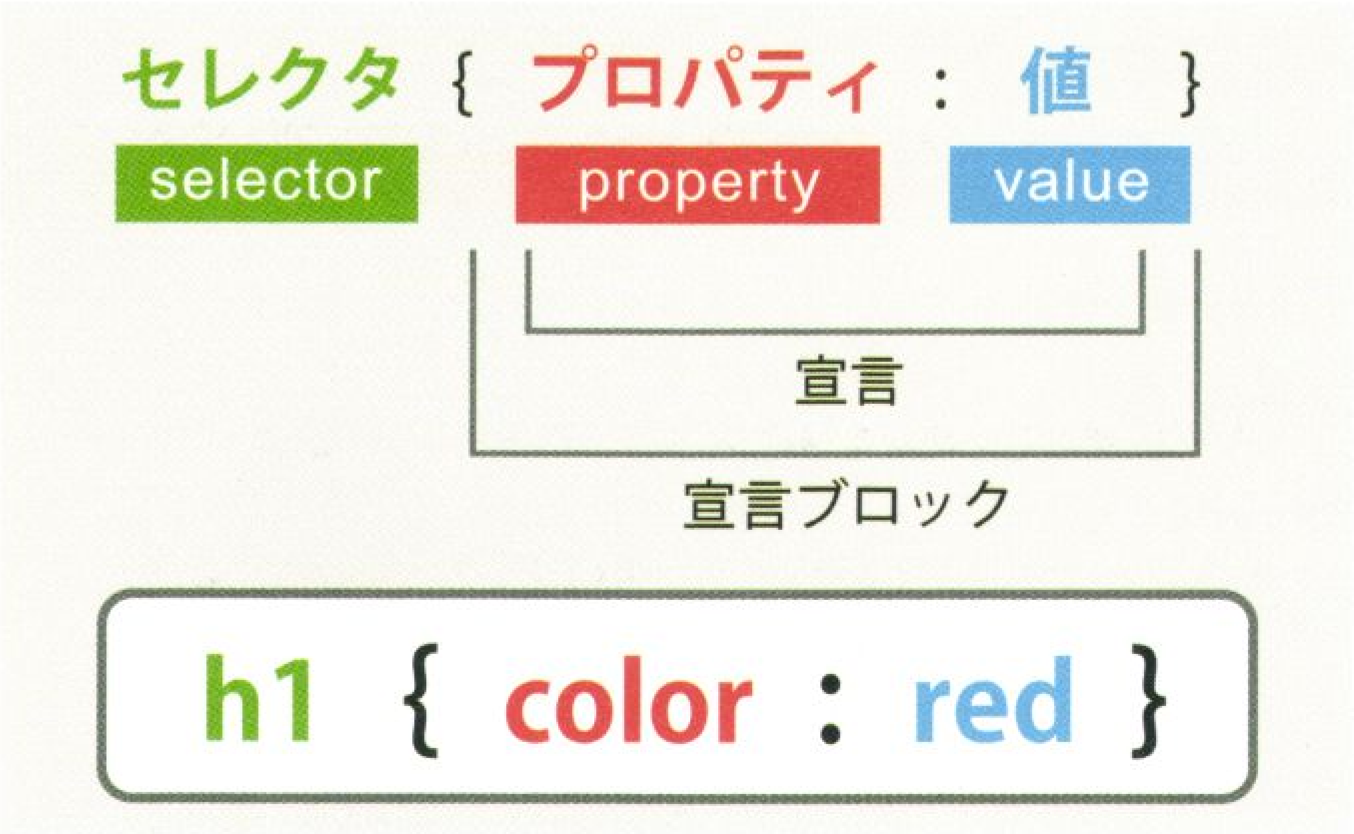
id属性とclass属性

特定の要素を識別しなければいけない場合は、id属性とclass属性を使います。

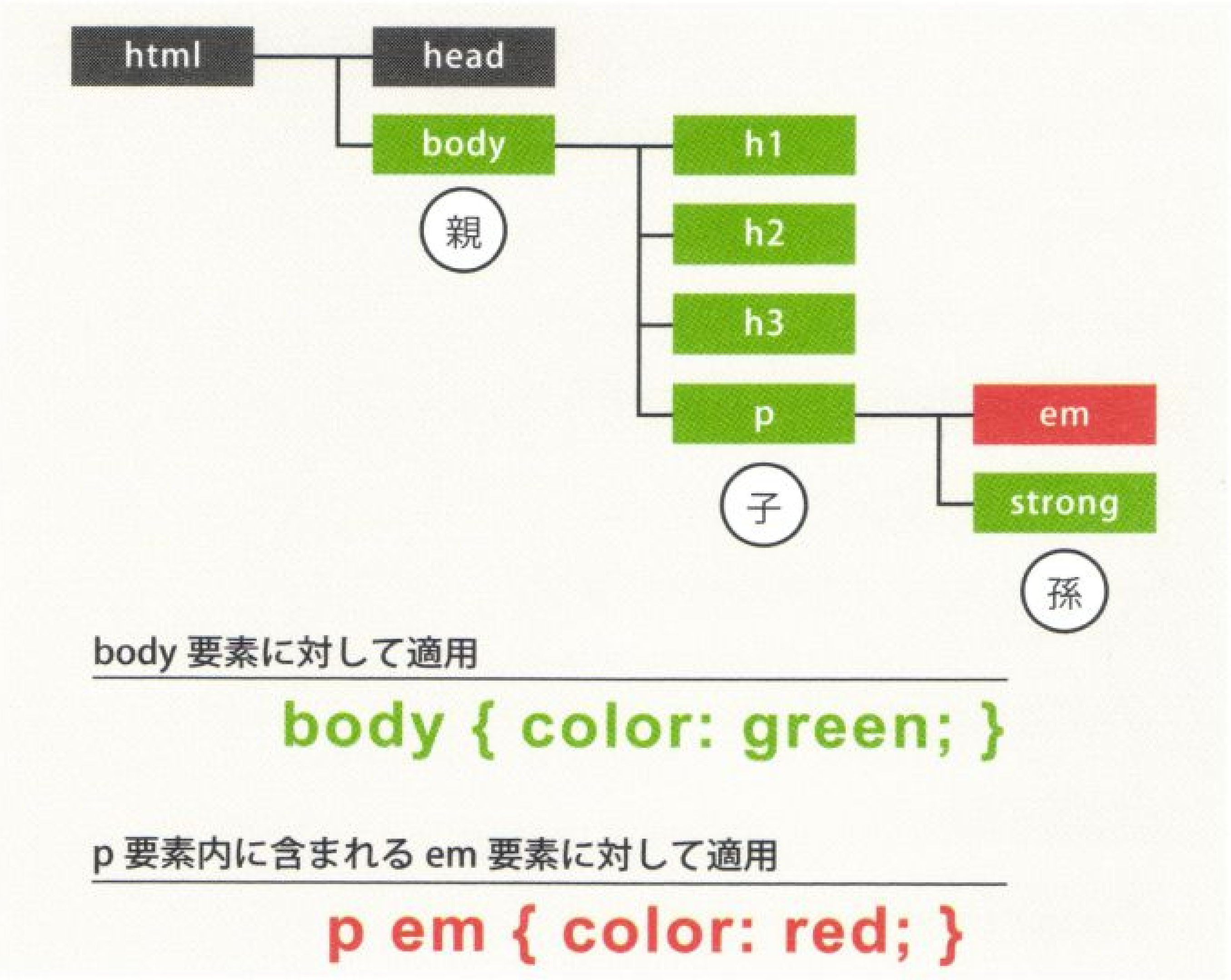
例えば、1つの段落だけにスタイルを適用したいときは、<p class="col">記事の本文</p>のように、class名を付けておけば、「p.col { color: red; }」と指定するだけで、colというクラスが記述されている段落のみ赤く表示されます。

id属性の場合も同様に、「<div id="hcol"><h3>記事の見出し</h3></div>」のように記述しておけば、「div#hcol { color: red; }」といったスタイルを適用することができます（この場合は記事の見出しだけ赤くなります）。1ページ内で1回しか使用されないページタイトルなどはid属性、記事の小見出しなど1ページ内で複数回使う可能性があるスタイルについてはclass属性を使いましょう。

06 CSSの基本文法

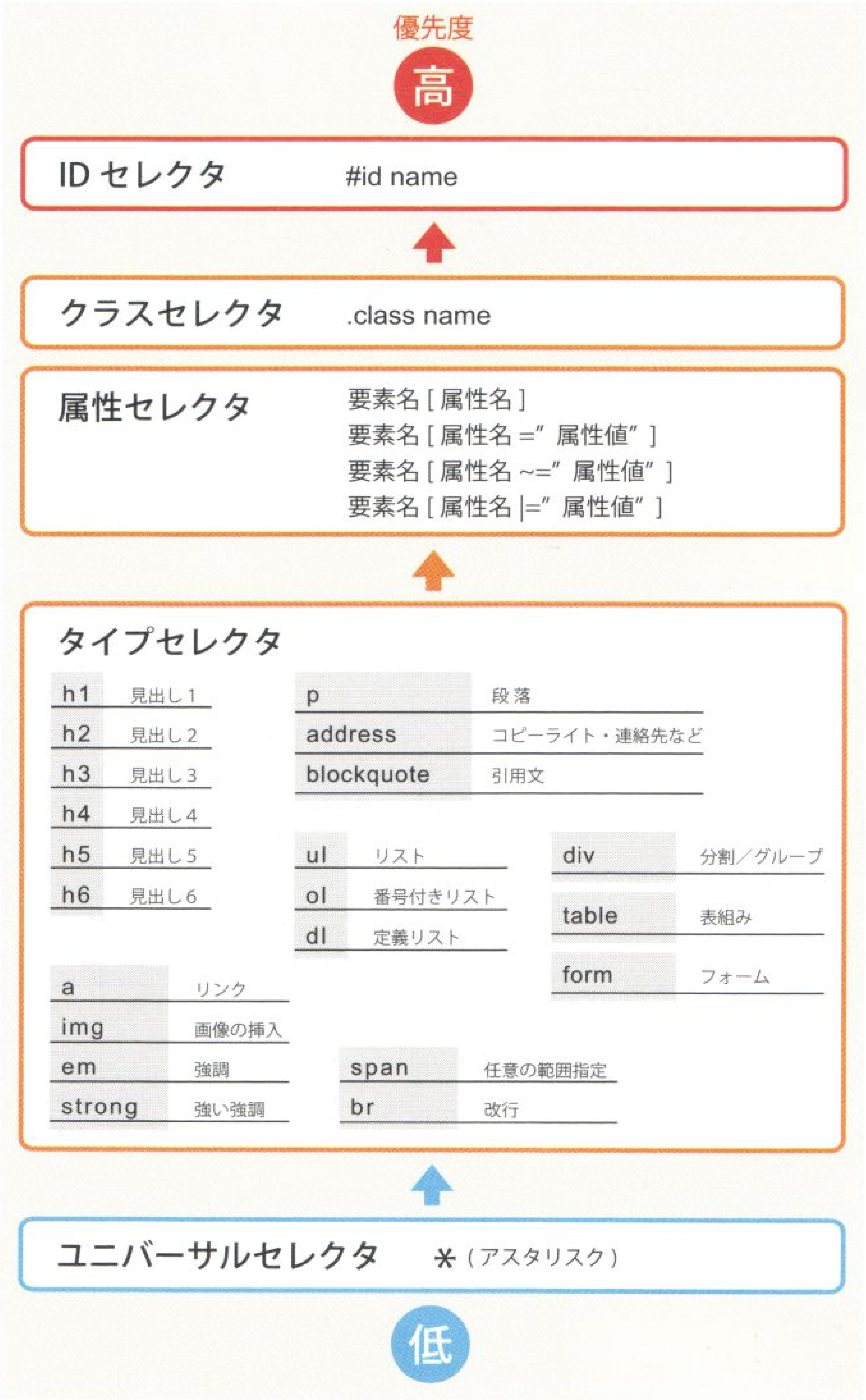


07 セレクタで指定したスタイルの「継承」の仕組み



指定したスタイルが、親要素から子要素に引き継がれる。「body { color:green; }」と「p em { color:red; }」を同時に指定すると、段落内のem要素だけ文字色が赤（color:red）になり、それ以外はすべて緑（color:green）になる

08 セレクタの種類と優先順位



ベーシックなセレクタには、タイプセレクタ、IDセレクタ、クラスセレクタ、属性セレクタなどがある。スタイルの指定が競合した場合、優先度によって適用される

ブラウザとベンダープレフィックス

現在普及しているデスクトップ環境のWebブラウザには、MicrosoftのInternet Explorer（IE）、Mozilla Firefox、Apple Safari、Google Chrome、Operaなどがあります。スマートデバイス（スマートフォン、タブレットなど）では、iOS環境のMobile SafariとAndroid環境のAndroid Browserが、主要なブラウザとなっています。

ブラウザには、HTMLやCSSを解釈して表示するためのレンダリングエンジンが搭載されており、IEはTrident（トライデント）、FirefoxはGecko（ゲッコー）、SafariとGoogle ChromeはWebKit（ウェブキット）、OperaはPresto（プレスト）が採用されています。レンダリング処理には差異があるため、同じページでもブラウザによって表示結果に違いが出てくる場合があります。

また、CSS3の先行実装には、ブラウザベンダーごとに異なる接頭辞（ベンダープレフィックスと呼びます）が用意されていますので、併記しなければいけません 12 。IEは「-ms-」、Firefoxは「-moz-」、SafariとChromeは「-webkit-」、Operaは「-o-」をプロパティの頭に付けます 13 。

09 「タイプセレクタ」以外のセレクタ

その他のセレクタ

子孫セレクタ	selector selector	
子セレクタ	selector > selector	
隣接セレクタ	selector + selector	

擬似クラス

リンク擬似クラス	:link :visited	初期状態（テキストリンク） リンク先を参照した状態
ダイナミック擬似クラス	:hover :active :focus	カーソルが重なっている状態 クリックしたときの状態 フォーカスされた状態
:before/:after 擬似クラス	:before :after	
:first-child 擬似クラス	:first-child	
:first-line 擬似クラス	:first-line	
:first-letter 擬似クラス	:first-letter	
言語擬似クラス	:lang()	

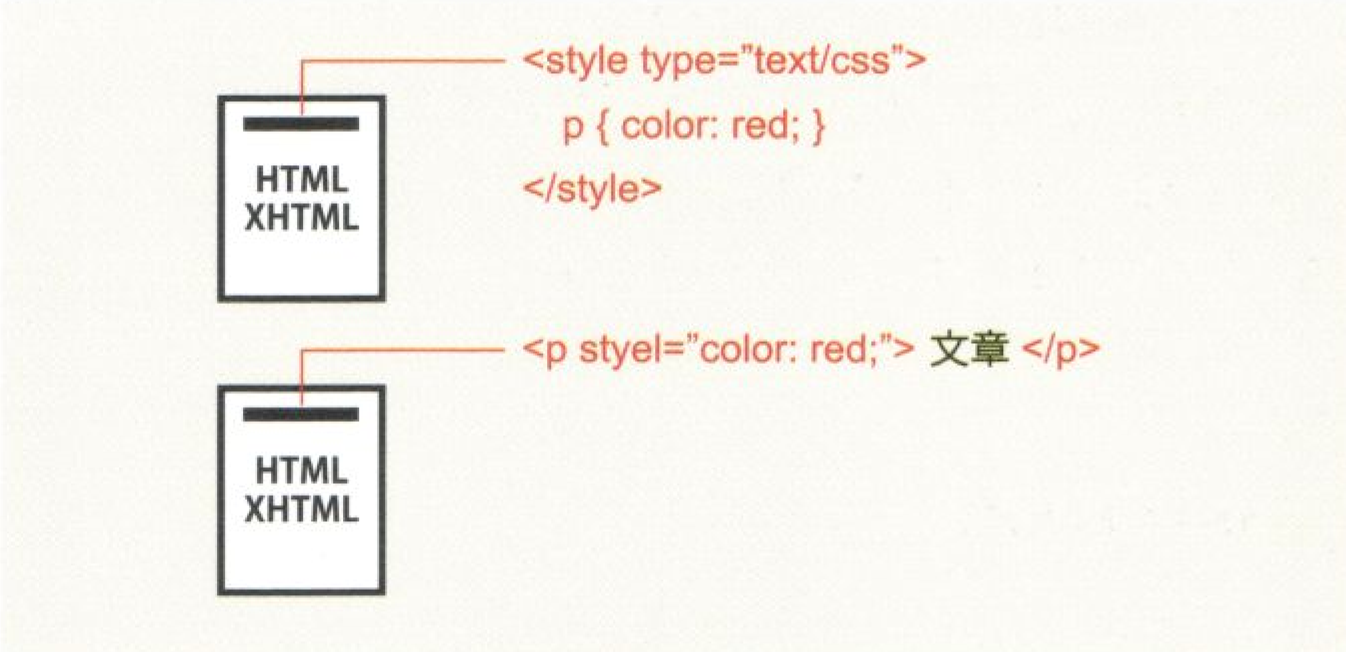
より効率よく指定するためのセレクタとして、子孫セレクタや子セレクタ、隣接セレクタなどが用意されている。また、特定の状態のとき（参照済みのページリンクの色を変更する等）にスタイルを指定できる、擬似クラスなどがある

10 外部ファイルにスタイルを記述し、HTMLで読み込む方法



CSSファイルを読み込む指定は、head要素内に記述する。link要素を使う方法が一般的

11 HTMLファイル内に直接記述する方法



head要素内に記述、もしくはHTML要素のstyle属性に直接記述する方法がある

12 ブラウザごとに異なる接頭辞（ベンダープレフィックス）



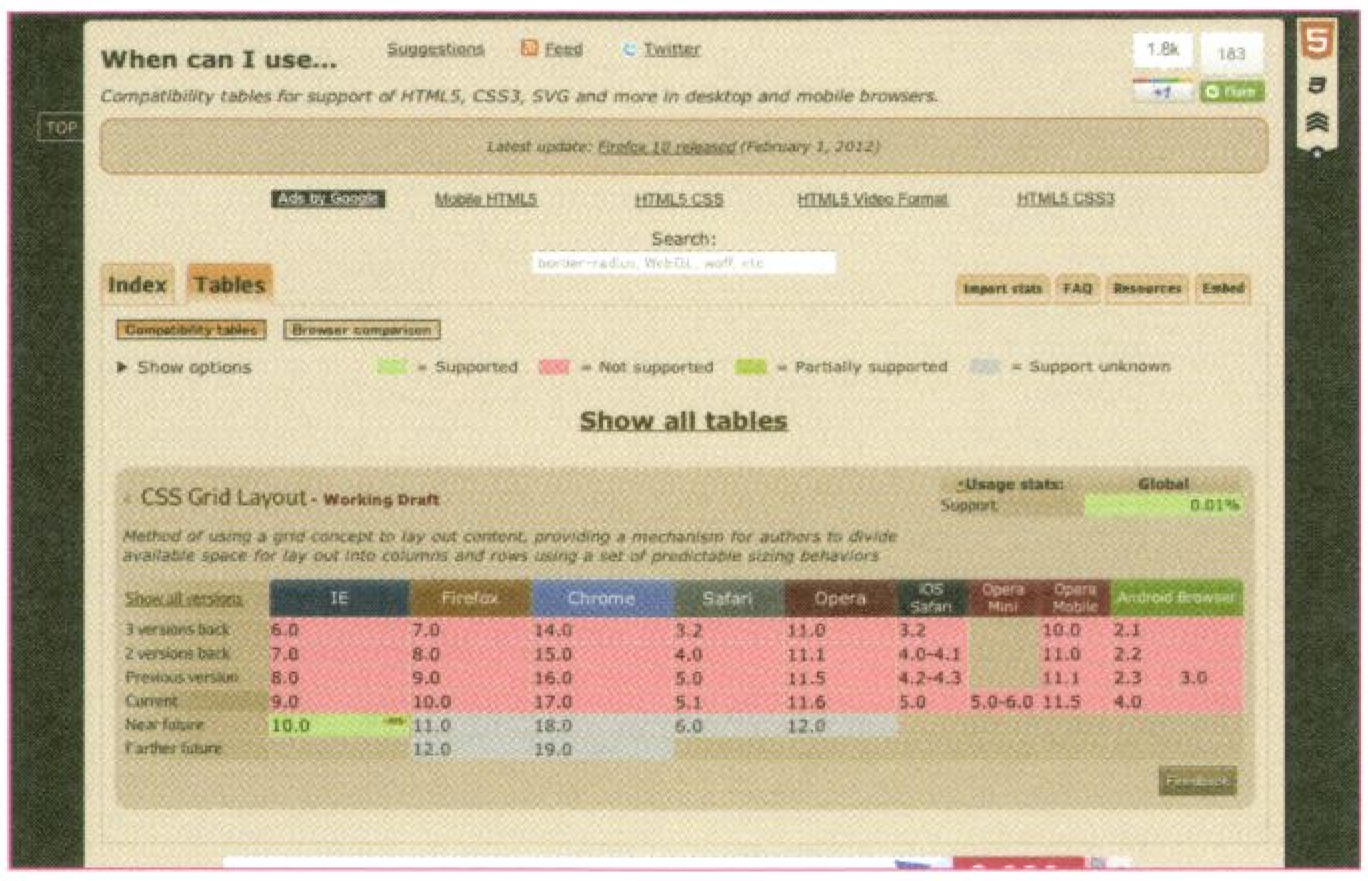
13 ベンダープレフィックスの記述例

```
-webkit-column-count: 2; /* Safari and Chrome */
-moz-column-count: 2; /* Firefox */
column-count: 2;
```

CSS3使用時の設計指針と注意点

CSS3はまだ策定中ですが、すでに先行実装されている仕様もあります。デスクトップ環境では、Internet Explorer（IE）、Firefox、Safari、Google Chrome、Operaの最新バージョン、スマートデバイスはiOSのMobile Safari、Androidの標準ブラウザなどが、CSS3の一部の仕様を試験的に実装しています。注意しなければいけないのは、ブラウザごとの実装状況の確認と未対応のブラウザに対する処置です。実装状況はブラウザのアップデートによって変化していきますので、定期的なチェックが欠かせません。ネットでも複数のサイトが情報を公開していますので、参考にしましょう。

ブラウザの実装状況が一覧できるサイト



When can I use...（http://caniuse.com/）

モバイルブラウザの実装状況が一覧できるサイト

Mobile HTML5 - compatibility tables for iPhone, Android, BlackBerry, Symbian, iPad and other mobile devices

http://mobilehtml5.org/

先行実装されているCSS3の一部の仕様を採用する場合は、サイトの方針を明確にしておくことが重要です。「プログレッシブ・エンハンスメント」、「クロスブラウザ」などの選択肢があります。

プログレッシブ・エンハンスメント

現在普及している閲覧環境を基準に設計した上で、CSS3などの最新の技術を先行実装したブラウザ向けにリッチなコンテンツを提供するという考え方です。

なお、「プログレッシブ・エンハンスメント」と逆に、先行実装された最新のブラウザを基準に設計し、古いブラウザなどは品質を落として対応するという選択肢もあります。この考え方が「グレイスフル・デグラデーション」です。「グレイスフル・デグラデーション」の場合には、IE6や7などの古いブラウザも切り捨てず、アクセシビリティを損なわないように対応しますが、見た目については考慮しません。

クロスブラウザ

先行実装された最新のブラウザを基準に設計し、古いブラウザに対してもスクリプトを駆使して、新しい環境に近づけるという考え方です。「ポリフィル（Polyfill）」と呼ばれ、JavaScriptのライブラリなどが提供されています。古いブラウザを力業で最新環境に近づけるため、処理が重たくなりますので動作検証は必須となります。

主なJavaScritpライブラリには以下のようなものがあります。

HTML5.js

HTML5の新要素をサポートしていないIE8以下をスクリプトで対応させるJavaScriptライブラリ。

指定方法：

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

HTML5 enabling script

http://remysharp.com/2009/01/07/html5-enabling-script/

IE9.js

IE5.5～8をIE9の実装に近づけるためのJavaScriptライブラリ。CSS3のセレクタが使用可能になり、透過PNGファイルも扱えるようになります。便利なライブラリですが動作検証は必須。

指定方法：

```
<!--[if lt IE 9]>
<script src="http://ie7-js.googlecode.com/svn/version/2.1(beta4)/IE9.js"></script>
<![endif]-->
```

ie7-js

http://code.google.com/p/ie7-js/

Respond.js

IE8以下でCSS3のメディアクエリ（Media Queries）を利用可能にするJavaScriptライブラリ。ライブラリを設置し、「<script src="../respond.src.js"></script>」のように記述するだけです。

Respond

https://github.com/scottjehl/Respond

CSS3で追加されたセレクタ

CSS3では要素の属性や状態などで選択できる新しいセレクタなどが加わったことで、例えば段ごとに背景色を設定するような指定など、CSS2.1には備わっていなかった高度なセレクタの組み合わせが可能になりました。HTMLの構造がシンプルかつセマンティックになり、CSSの肥大化も抑えることができます。

セレクタ

CSSのスタイルは、「セレクタ { プロパティ : 値 ; }」のように、セレクタを使って適用します。

セレクタには、タイプセレクタやユニバーサルセレクタ、クラスセレクタ、IDセレクタ、属性セレクタ、子孫セレクタ、子セレクタ、隣接セレクタ、間接セレクタ、擬似クラス、擬似要素などがあります。CSS3では、「Selectors Level3」というモジュールで新しいセレクタが定義されており 01 、ほかにも「テキスト・フォント」、「レイアウトデザイン」、「エフェクト」などの機能別にモジュールが用意されています 02 。

CSS3で追加された属性セレクタ

属性セレクタは、要素の属性とその属性の値をもとに、対象を選択するセレクタです 03 。

E[foo^="bar"]

foo（属性名）で指定した値が、"bar"で始まる場合に適用されます。

```
object[type^="../image/"] { スタイルの指定 }
```

E[foo$="bar"]

foo（属性名）で指定した値が、"bar"で終わる場合に適用されます。

```
a[href$=".html"] { スタイルの指定 }
```

E[foo*="bar"]

foo（属性名）で指定した値に、"bar"が含まれる場合に適用されます。

```
p[title*="hello"] { スタイルの指定 }
```

01 「Selectors Level 3」

http://www.w3.org/TR/css3-selectors/

02 機能別カテゴリとCSS3のモジュール

- セレクタ
  - Selectors Level 3
- テキスト・フォント
  - CSS Fonts Module Level 3
  - CSS Text Level 3
  - CSS Writing Modes Module Level 3
  - etc
- レイアウトデザイン
  - CSS Multi-column Layout Module
  - CSS Flexible Box Layout
  - CSS Grid Layout
  - etc
- エフェクト
  - CSS 2D Transforms
  - etc
- カラー・イメージ
  - CSS Color Module Level 3
  - etc
- リスト
  - CSS Lists and Counters Module Level 3
  - etc
- その他

03 属性セレクタの一覧

属性セレクタ	適用対象
E[foo^="bar"]	an E element whose "foo" attribute value begins exactly with the string "bar" 属性「foo」の値が「bar」で指定された文字列から始まる場合に適用
E[foo$="bar"]	an E element whose "foo" attribute value ends exactly with the string "bar" 属性「foo」の値が「bar」で指定された文字列で終わる場合に適用
E[foo*="bar"]	an E element whose "foo" attribute value contains the substring "bar" 属性「foo」の値に「bar」で指定された文字列が含まれる場合に適用

http://www.w3.org/TR/css3-selectors/#attribute-selectors

CSS3で追加された主な擬似クラス

擬似クラスは、各要素の状態によって選択するセレクタです 04。各要素が特定の条件を満たした状態のときに、スタイルを適用します。主な擬似クラスの記述方法は以下の通りです。

E:root

ルート要素（HTMLではhtml要素）に適用されます。

```
:root { スタイルの指定 }
```

E:nth-child(n)

同じ親要素を持つ兄弟の要素で、前から何番目もしくは奇数・偶数番目の要素に適用されます。たとえば、「2n+1」の場合は、前から1番目（2×0＋1=1)、3番目（2×1＋1=3)、5番目（2×2＋1=5）……のように、奇数番目の要素が対象になります。「odd」も奇数番目に適用します。偶数番目は「even」を使います。

・記述例A（奇数行の要素に適用）

```
tr:nth-child(2n+1) { スタイルの指定 } ／ tr:nth-child(odd) { スタイルの指定 }
```

・記述例B（偶数行の要素に適用）

```
tr:nth-child(2n+0) { スタイルの指定 } ／ tr:nth-child(even) { スタイルの指定 }
```

E:nth-last-child(n)

同じ親要素を持つ兄弟の要素で、後ろから何番目もしくは奇数・偶数番目の要素に適用されます。

・記述例A（後ろから奇数行の要素に適用）

```
tr:nth-last-child(2n+1) { スタイルの指定 } ／ tr:nth-last-child(odd) { スタイルの指定 }
```

・記述例B（後ろから偶数行の要素に適用）

```
tr:nth-last-child(2n+0) { スタイルの指定 } ／ tr:nth-last-child(even) { スタイルの指定 }
```

E:nth-of-type(n)

同じ親要素を持つ同一の要素で、前から何番目もしくは奇数・偶数番目の要素に適用されます。記述例の場合、「img」が指定されていますので、同じ親要素を持つ兄弟の要素の中で、img要素だけが対象となります。

```
img:nth-of-type(2n+1) { スタイルの指定 }
```

E:nth-last-of-type(n)

同じ親要素を持つ同一の要素で、後ろから何番目もしくは奇数・偶数番目の要素に適用されます。

```
img:nth-last-of-type(2n+1) { スタイルの指定 }
```

04 擬似クラスの一覧

擬似クラス	適用対象
E:root	an E element, root of the document 文書のルート要素に適用
E:nth-child(n)	an E element, the n-th child of its parent 同じ親要素を持つ兄弟の要素で、前から一定間隔の要素に適用
E:nth-last-child(n)	an E element, the n-th child of its parent, counting from the last one 同じ親要素を持つ兄弟の要素で、後ろから一定間隔の要素に適用
E:nth-of-type(n)	an E element, the n-th sibling of its type 同じ親要素を持つ兄弟の要素で、前から一定間隔の同一要素に適用
E:nth-last-of-type(n)	an E element, the n-th sibling of its type, counting from the last one 同じ親要素を持つ兄弟の要素で、後ろから一定間隔の同一要素に適用
E:first-child	an E element, first child of its parent 同じ親要素を持つ兄弟の要素で、最初の要素に適用
E:last-child	an E element, last child of its parent 同じ親要素を持つ兄弟の要素で、最後の要素に適用
E:first-of-type	an E element, first sibling of its type 同じ親要素を持つ兄弟の要素で、最初の同一要素に適用
E:last-of-type	an E element, last sibling of its type 同じ親要素を持つ兄弟の要素で、最後の同一要素に適用
E:only-child	an E element, only child of its parent １つしかない子要素に適用
E:only-of-type	an E element, only sibling of its type 同じ親要素を持つ兄弟の要素で、１つしかない子要素に適用
E:empty	an E element that has no children (including text nodes) 子要素がない空の要素に適用

http://www.w3.org/TR/css3-selectors/#structural-pseudos

CSS3で追加されたターゲット擬似クラス

アンカーリンクの対象となる要素に適用する擬似クラスが「ターゲット擬似クラス」(:target) です 05。

E:target

リンクのターゲットに適用されます。たとえば、「<p id="note">テキスト</p>」のようにID属性を記述しておけば、「クリック」のテキストリンクをクリックすると、p要素にスタイルの指定が適用されます。

```
#note:target { スタイルの指定 }
```

このほかに、UI要素（ラジオボタンやチェックボックスなど）の状態を判定して適用する「UI要素状態擬似クラス」 06、指定したセレクタ以外のセレクタに適用される「否定擬似クラス」 07、要素Eのあとに記述されている要素に対して適用される「一般兄弟結合子」 08 などがあります。

一般兄弟結合子

チルダ（~）を結合子にします。記述例では、h1要素の後に記述されている（同階層の）全てのpre要素にスタイルが適用されます。

```
<h1>見出し</h1>
<p>記事の解説文</p>
<pre>整形済みのテキスト</pre>

h1 ~ pre { スタイルの指定 }
```

05 ターゲット擬似クラス

ターゲット擬似クラス	適用対象
E:target	an E element being the target of the referring URI リンクのターゲットに適用

http://www.w3.org/TR/css3-selectors/#target-pseudo

06 UI要素擬似クラス

UI要素擬似クラス	適用対象
E:enabled E:disabled	a user interface element E which is enabled or disabled disabled属性がない要素(:enabled)、もしくはdisabled属性が指定されている要素(:disabled)に適用
E:checked	a user interface element E which is checked (for instance a radio-button or checkbox) checked属性がある要素(checked状態のチェックボックスなど)に適用

http://www.w3.org/TR/css3-selectors/#UIstates

07 否定擬似クラス

否定擬似クラス	適用対象
E:not(s)	an E element that does not match simple selectors 指定したセレクタと一致しない要素に適用

http://www.w3.org/TR/css3-selectors/#negation

08 一般兄弟結合子

一般兄弟結合子	適用対象
E ~ F	an F element preceded by an E element 要素「E」の後に記述されている要素「F」に適用

http://www.w3.org/TR/css3-selectors/#general-sibling-combinators

CSS3で追加されたプロパティ

CSS3で新しく追加されたプロパティを利用することで、文字組の指定、イメージ（画像）を使用しないと表現できなかったビジュアル効果などが実現可能になりました。テキストとフォント、エフェクト、カラーとイメージなどに分けて、主な仕様について解説していきましょう。

テキストとフォント

CSS3では、テキストやフォントの仕様が大幅に拡張されています。フォント関連の仕様「CSS Fonts Module Level 3」01、テキスト関連の仕様「CSS Text Level 3」02、組み方向などの仕様「CSS Writing Modes Module Level 3」03などがあり、Webフォントの利用や禁則処理、改行の調整、下線や圏点の調整、ドロップシャドウ、縦書きなど、テキストとフォントの表現力が向上しています。

01 「CSS Fonts Module Level 3」のプロパティ一覧（フォント名、文字サイズ、スタイル等の指定）

プロパティ	指定内容
font	フォントについての指定をまとめて行う
font-family	フォントの種類を指定
font-feature-settings	OpenTypeフォント機能の有効・無効の切り替え、装飾の指定
font-kerning	カーニングについての指定
font-language-override	言語の種類を切り替える指定
font-size	フォントのサイズを指定
font-size-adjust	フォントのサイズ(アスペクト比)を調整
font-stretch	フォントを縦長・横長にする
font-style	フォントスタイル(標準・イタリック・斜体)を指定する
font-synthesis	フォント合成の指定
font-variant	フォントをスモールキャプスで表示する
font-variant-alternates	修飾文字の指定
font-variant-caps	スモールキャプスに切り替える指定
font-variant-east-asian	異体字の指定
font-variant-position	上付き・下付きの指定
font-weight	フォントの太さを指定
font-variant-ligatures	合字(リガチャ)の指定
font-variant-numeric	分数表示の指定

http://www.w3.org/TR/css3-fonts/#property-index

02 「CSS Text Level 3」のプロパティ一覧 改行や行揃え、禁則処理などの組版ルールの指定

プロパティ	指定内容
hanging-punctuation	句読点のぶら下げ組みの指定
hyphens	単語の分割(ハイフネーション)
letter-spacing	文字の間隔の指定
line-break	禁則処理の指定
overflow-wrap	単語の改行についての指定
tab-size	タブの幅の指定
text-align	行揃えの指定
text-align-last	最終行の行揃えの指定
text-decoration	文字の線(下線・上線・取り消し線)の指定
text-decoration-color	文字の線の色を指定
text-decoration-line	文字の線の種類を指定
text-decoration-skip	線を非表示にしたい部分(スペース、タブなど)の指定
text-decoration-style	文字の線のスタイルを指定
text-emphasis	圏点を表示する指定
text-emphasis-color	圏点の色の指定
text-emphasis-position	圏点の配置の指定
text-emphasis-style	圏点のスタイルの指定
text-indent	最初の行の字下げ(インデント)の指定
text-justify	両端揃え指定の調整
text-shadow	テキストの影(ドロップシャドウ)の指定
text-space-collapse	空白文字の処理を適用するかどうか指定
text-transform	アルファベットの大文字・小文字の指定
text-underline-position	アンダーライン(下線)の配置指定
text-wrap	テキストの折り返しの指定
white-space	空白文字(スペースやタブ、改行)の処理についての指定
word-break	文章の改行についての指定
word-spacing	単語の間隔の指定

http://www.w3.org/TR/css3-text/#appendix-h-full-property-index

Webフォントの指定

Webフォントは、サーバー内にあるフォントデータのURLとフォーマットを指定して読み込む仕組みです。Webフォントを利用することで、ユーザーのローカル環境にインストールされていないフォントを使ってWebサイトを表示することができます（※あらかじめフォントのライセンスについて確認しておく必要があります）。フォーマットには、WOFF（Web Open Font Format）やTrueType、OpenType、SVG Font、そして、IEがサポートしているEOT（Embedded OpenType ）などがあります。

Webフォントを読み込むには、@font-face規則（The @font-face rule）を使って、ソースとなるURLやフォントの種類を記述します。以下は指定できるプロパティと記述例です 04 05。

文字の大きさの自動調整

Webページで指定されているフォントが閲覧しているデバイスにない場合、代替フォントが表示されますが、文字サイズのバランスなども変わってしまいます。

font-size-adjustプロパティで、数値（縦横比）を指定すると、自動調整してくれます（小文字「x」が同じ高さになるように調整されます）。

font-size-adjust

初期値　none
適用要素　すべての要素
継承　あり
値　<number> | none | inherit

```
h1 {
  font-size-adjust: 0.5;
}
```

禁則処理の指定

文章の見栄えを整える禁則処理（行頭に「。」や「、」、小文字「っ」などが表示されない等）のレベルについて指定します。値には、auto（ブラウザのデフォルト）、loose（ゆるい）、normal（標準）、strict（厳密）があります。

line-break

初期値　auto
適用要素　すべての要素
継承　あり
値　auto | loose | normal | strict

```
p {
  -ms-line-break:normal;
  line-break:normal;
}
```

03 「CSS Writing Modes Module Level 3」のプロパティ一覧（縦書き・横書きなどの指定）

プロパティ	指定内容
direction	書字方向の指定
'caption-side'	表組みのキャプションの配置
'column-width'	カラムの幅
'width', 'min-width',	幅の値　幅の最小値
'max-width', 'height',	幅の最大値　高さの値
'min-height', 'max-height'	高さの最小値　高さの最大値
text-combine-horizontal	縦組みのグリフの水平合成
text-combine-mode	縦組みの水平方向のグリフ・スケーリング
text-orientation	縦書きテキスト内の文字の回転
unicode-bidi	言語ごとの書字方向ルールの変更
writing-mode	組み方向（縦書き・横書き）の指定

http://www.w3.org/TR/css3-writing-modes/#property-index

04 @font-faceで指定できるプロパティ

プロパティ	指定する内容
font-family	フォントファミリー
src	フォントのソース
font-style	フォントのスタイル
font-weight	フォントの太さ
font-stretch	フォントの横幅
unicode-range	グリフのコードの範囲

http://www.w3.org/TR/css3-fonts/#font-face-rule

05 @font-faceの記述例

```
@font-face {
  font-family: Gentium;
  src: url(http://example.com/fonts/Gentium.ttf);
}
```

自動改行

自動的に改行されるルールについて指定します。値には、normal（日本語、中国語、韓国語は改行され、その他は言語のルールで決まる）、keep-all（日本語、中国語、韓国語は単語の途中で改行されない、その他は言語のルールで決まる）、break-all（日本語、中国語、韓国語以外は、単語の途中でも改行される）があります。

word-break

初期値	normal
適用要素	すべての要素
継承	あり
値	normal | keep-all | break-all

```
p {
  -ms-word-break:break-all;
  word-break:break-all;
}
```

両端揃え

text-alignプロパティで「justify」（両端揃え）を指定した際、言語ごとの規則を指定することができます。

値には、auto（ブラウザのデフォルト）、none（なし）、inter-word（スペースの間隔で調整）、inter-ideograph（スペースの調整と、日本語や中国語などは字間で調整）、inter-cluster（スペースの調整と、タイ語やミャンマー語などは字間で調整）、distribute（スペースの調整と字間の調整。アラビア語やベンガル語などは除く）、kashida（アラビア語などスペースや字間で調整できない言語を引き伸ばすことで調整）があります。

text-justify

初期値	auto
適用要素	ブロック要素（オプションとしてインライン要素も適用可能）
継承	あり
値	auto | none | inter-word | inter-ideograph | inter-cluster | distribute | kashida

```
p {
  -ms-text-justify:inter-ideograph;
  text-justify:inter-ideograph;
}
```

最終行の行揃え

文章の最後の行に対して、行揃えを指定します。値には、auto（text-alignプロパティの値）、start（書字方向の開始）、end（書字方向の終端）、left（左）、right（右）、center（中央）、justify（両端）があります。

text-align-last

初期値	auto
適用要素	ブロック要素
継承	あり
値	auto | start | end | left | right | center | justify

```
p {
  -ms-text-align-last:right;
  text-align-last:right;
}
```

下線の位置

text-decorationプロパティで、指定された下線の位置を決めることができます。値には、auto（ブラウザのデフォルト）、alphabetic（ベースライン）、below left（テキストの下に表示、縦書きでは左に表示）、below right（テキストの上に表示、縦書きでは右に表示）があります。

IEは、独自拡張のbelow（below leftと同じ）とabove（below rightと同じ）を使用します。

text-underline-position

初期値	auto
適用要素	すべての要素
継承	あり
値	auto | alphabetic | [below || [left | right]]

```
p {
  -ms-text-underline-position: below;
  text-underline-position: below left;
}
```

※ IEでは、below（独自拡張）でbelow leftが適用されます。IE7以下を対象とする場合は、ベンダープレフィックス「-ms-」を付けません。

圏点

文章の一部を「、」や「・」などの記号で強調する「圏点」を指定します。「text-emphasis: スタイル 色」で記述します。スタイルには、dot（ドット）、circle（丸）、double-circle（二重丸）、triangle（三角）、sesame（ゴマ）があり、filled（塗り）もしくはopen（白抜き）が指定できます。

text-emphasis

初期値	none
適用要素	すべての要素
継承	あり
値	'<text-emphasis-style>' || '<text-emphasis-color>'

```
span {
  -webkit-text-emphasis:sesame red;
  text-emphasis:sesame red;
}
```

圏点の位置

text-emphasisプロパティで、指定された圏点の位置を決めることができます。値には、above right（テキストの上、縦書きの場合は右）、below right（テキストの下、縦書きの場合は右）などがあります。

text-emphasis-position

初期値	above right
適用要素	すべての要素
継承	あり
値	[above | below] && [right | left]

```
span {
  -webkit-text-emphasis-position:over;
  text-emphasis-position:above right;
}
```

Chromeでは、above rightが「over」、below rightが「under」で適用されます。

ドロップシャドウ

テキストにドロップシャドウ（影）を適用することができます。「text-shadow: 水平方向のオフセット値　垂直方向のオフセット値　陰影のぼかしの半径（省略可）　色」のように記述します。水平方向と垂直方向のオフセット値、ぼかし半径、色の組み合わせをセットにして、カンマ（,）で区切れば、複数指定することが可能です。

text-shadow

初期値	none
適用要素	すべての要素
継承	あり
値	none | [<length>{2,3} && <color>?]#

```
h1 {
  text-shadow:3px 3px 4px #999;
}
```

縦書きと横書き

文章の書字方向（横書き・縦書き）を指定します。値には、horizontal-tb（横書きで行が上から下へ流れる）、vertical-rl（縦書きで列が右から左へ流れる）、vertical-lr（縦書きで列が左から右へ流れる）があります。縦書きの場合、Windows環境では、縦書き用のフォント名（日本語のフォント名の頭に「@」）を指定する必要があります。

なお、IEの場合は、独自拡張のtr-tb（horizontal-tbと同じ）、tb-rl（vertical-rlと同じ）、tb-lr（vertical-lrと同じ）で対応することになります。

writing-mode

初期値	horizontal-tb
適用要素	すべての要素（適用されない：table row groups, table column groups, table rows, table columns）
継承	あり
値	horizontal-tb | vertical-rl | vertical-lr

```
html {
  -webkit-writing-mode:vertical-rl;
  writing-mode:vertical-rl;
}
```

エフェクト

アニメーションやトランジションなどの特殊効果は、JavaScriptやFlashなどの拡張技術を使用していましたが、CSS3には「CSS 2D Transforms」や「CSS 3D Transforms Module Level 3」、「CSS Animations Module Level 3」などの仕様が追加されており、CSSだけでオブジェクトのアニメーションや3D効果、スライドショーのトランジション効果などを簡単に実行することが可能になりました 06。

今後は、Adobe Edge 07 などのHTML5／CSS3アニメーション制作のオーサリングツールが普及していく可能性があります。

transform

transformプロパティは、対象とする要素に対して、移動・回転・伸縮（拡大・縮小）・傾斜（シアー）などの2次元トランスフォーム処理を実行します。

移動はtranslate()、translateX()、translateY()、回転はrotate()、伸縮はscale()、scaleX()、scaleY()、傾斜はskew()、skewX()、skewY()を指定します（トランスフォーム関数と呼びます）。

初期値　none

適用要素　ブロックレベル要素、インライン要素

継承　なし

値　none | <transform-function> [<transform-function>]*

```
#obj {
  -moz-transform: translate(480px, 80px) scale(2, 2)
  rotate(120deg);
  -webkit-transform: translate(480px, 80px) scale(2, 2)
  rotate(120deg);
  -o-transform: translate(480px, 80px) scale(2, 2) rotate(120deg);
  transform: translate(480px, 80px) scale(2, 2) rotate(120deg);
}
```

カラーとイメージ

CSS2.1でボックスの角を丸くしたり、テキストに陰影をつけるには、イメージ（画像）を使用します。これらの視覚表現は、div要素を増加させ、CSSを複雑にしてしまうなど、アクセシビリティ上の問題やメンテナンス性の低下などが指摘されてきました。

CSS3には、「CSS Color Module Level 3」や「CSS Backgrounds and Borders Module Level 3」などの仕様が追加されており、角丸の指定やボックスのドロップシャドウ、線形および円形のグラデーション表示、複数の背景画像を重ねて表示する機能など、今まで画像で表現してきたビジュアルデザインをCSSだけで指定できるようになりました 08。

border-radius

border-radiusプロパティは、対象とするボックスの角を丸くします（角丸の半径の値を指定します）。値が1つの場合は、4つの角に適用され、2つの場合は左上と右下の角、右上と左下の角にそれぞれ適用されます。3つなら左上、右上と左下、右下の角、4つの場合は、左上から時計回りに適用されます。

初期値　0

適用要素　すべての要素（border-collapseの値がcollapseのtable要素は除く）

継承　なし

値　[<length> | <percentage>]{1,4} [/ [<length> | <percentage>]{1,4}]?

```
#contents {
  border: solid 3px #039;
  -moz-border-radius: 18px;
  -webkit-border-radius: 18px;
  border-radius: 18px;
}
```

リストマーク

リスト関連では、CSS2.0で定義されていた漢数字「一、二、三」やひらがな「あいうえお／いろは」、カタカナ「アイウエオ／イロハ」、ヘブライ数字などのリストマーク（CSS2.1では削除）が、CSS3で復活しています 09。

list-style

list-styleプロパティは、「list-style: リストマークの種類　イメージ（画像）表示する位置」のように指定します。値は省略可能です。CSS3では、CSS2.1で削除され、再び採用された漢数字（cjk-ideographic）やひらがな（hiragana）、カタカナ（katakana）などが使用できます。

初期値　disc none outside

適用要素　「display: list-item」の要素

継承　あり

値　<'list-style-type'> || <'list-style-position'> || <'list-style-image'>

```
li {
  list-style:cjk-ideographic;
}
```

06 アニメーション、トランジションなどエフェクト関連のモジュール

CSS 2D Transforms

http://www.w3.org/TR/css3-2d-transforms/

CSS Animations Module Level 3

http://www.w3.org/TR/css3-animations/

CSS 3D Transforms Module Level 3

http://www.w3.org/TR/css3-3d-transforms/

07 Adobe Edge

http://labs.adobe.com/technologies/edge/

08 カラー・イメージ関連のモジュール

CSS Color Module Level 3

http://www.w3.org/TR/css3-color/

CSS Backgrounds and Borders Module Level 3

http://www.w3.org/TR/css3-background/

09 CSS Lists and Counters Module Level 3

http://www.w3.org/TR/css3-lists/

レイアウトデザイン

CSS2.1での段組みレイアウトの表現は、floatプロパティで要素を浮動化して幅の値を与えることで、後続の要素を水平方向（左右どちらか）に並べていました。CSS3では段数を指定するなど、より直感的に表現できる機能が追加されており、最新の機能を組み合わせて指定すれば、高度なレイアウトも簡単に構築することが可能です。

マルチカラムレイアウト

「マルチカラムレイアウト」では、段数や段の幅、間隔、罫線などを指定することができます 01。段数はcolumn-countプロパティ、段の幅はcolumn-widthプロパティで指定します。たとえば、「column-count: 3;」と記述すると3段組みで表示されます。組方向が横の場合、左側のボックスから順に流し込まれます。

段の間隔は、column-gapプロパティで指定します。「column-gap: 32px;」と記述すると、段と段の間隔は32ピクセルになります。

段と段の間に表示する罫線（区切り線）は、column-ruleプロパティで指定します。「column-rule: 1px solid #f00;」と記述すると、太さ1ピクセルの単線が赤で表示されます 02。

マルチカラムレイアウトでは、ボックスの段の数をcolumn-countプロパティ（1以上の整数値を指定）を使用して指定します。段の幅を指定するcolumn-widthプロパティと組み合わせて使うことが可能です。

column-count

初期値	auto
適用要素	ブロックレベル要素（置換要素、table要素を除く）、th要素・td要素、インラインブロック要素
継承	なし
値	<integer> | auto

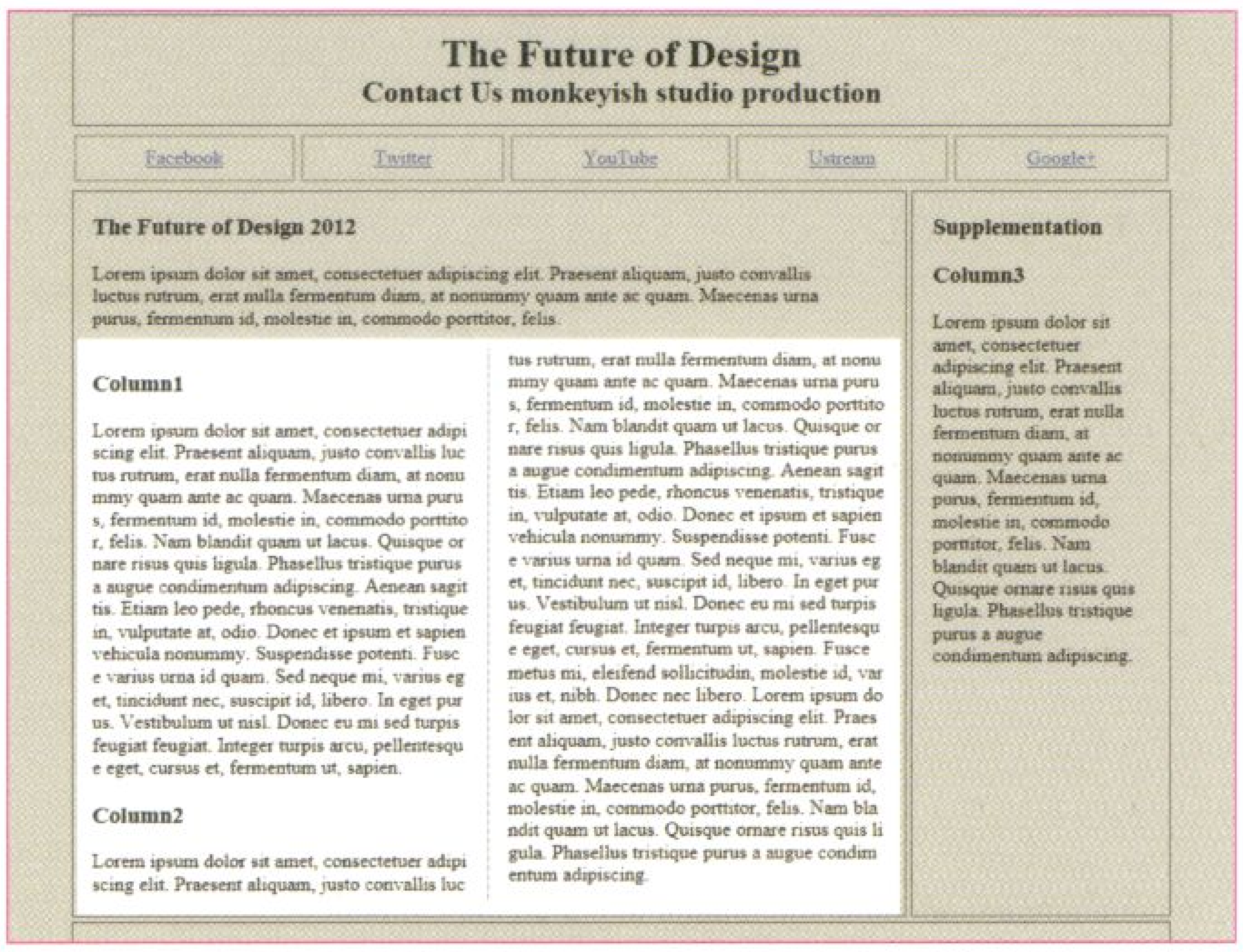
```
#contents {
  -moz-column-count:2;
  -webkit-column-count:2;
  column-count:2;
}
```

01 「CSS Multi-column Layout Module」のプロパティ一覧（マルチカラムレイアウトの指定）

プロパティ	指定内容
break-after	改ページ・改段の挿入位置（ボックスの前）の指定
break-before	改ページ・改段の挿入位置（ボックスの後）の指定
break-inside	改ページ・改段の挿入位置（ボックス内）の指定
column-count	段数の指定
column-fill	段の高さを揃える指定
column-gap	段と段の間隔の指定
column-rule	段の区切り線の指定
column-rule-color	区切り線の色の指定
column-rule-style	区切り線のスタイルの指定
column-rule-width	区切り線の太さの指定
columns	段の幅と段数の指定
column-span	複数の段を貫く表示の指定
column-width	段の幅の指定

http://www.w3.org/TR/css3-multicol/#property-index

02 記事に「CSS Multi-column Layout Module」の機能を適用



フレキシブルボックスレイアウト

マルチカラムレイアウトは、1つのボックスを分割して段組みを表現しますが、フレキシブルボックスレイアウトは、複数のボックスを水平方向に配置します（floatプロパティを使った段組みの手法に近い仕様です） 03 04 。

複数のボックスを含有する親ボックスに対して「display: box;」と指定すれば（※IEのベンダープレフィックス「-ms-」を付けます）、ボックスが水平方向に並びます。ボックスの高さを自動的に揃えてくれますので、意図しない空白が発生してバランスが崩れることもありません。

ボックスにdisplayプロパティを使用し、「flexbox」を指定することでブロック要素として扱われ、フレキシブルボックスが有効になります。デフォルトでは、水平方向（左から右）に並びますが、box-orientプロパティを使えば、垂直方向に並べることができます。

display

初期値　inline

適用要素　すべての要素

継承　なし

値　flexbox | inline-flexbox（※box | inline-box から変更されているがベンダープレフィックスは以前のまま）

```
nav {
  display: -moz-box;
  display: -webkit-box;
  display: -ms-box;
  display: flexbox;
  width: 200px;
}
```

グリッドレイアウト

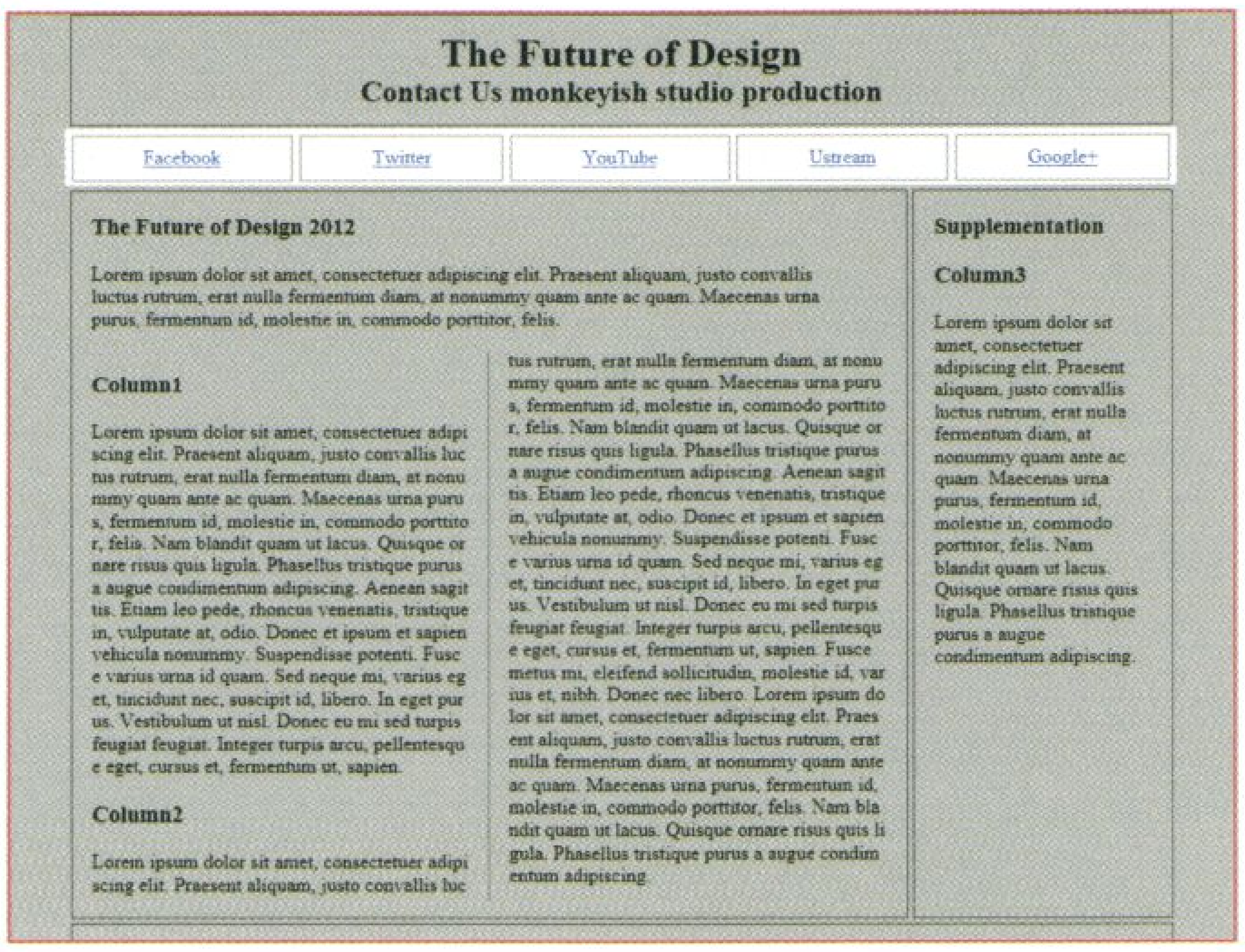
「グリッドレイアウト」では、適用したい領域に対して「display: grid」を指定し（※IEのベンダープレフィックス「-ms-」を付けます）、グリッドコンテナとして定義してお

03 「CSS Flexible Box Layout Module」のプロパティ一覧（フレキシブルボックスレイアウトの指定）

プロパティ	指定内容
display	ボックス（flexbox）の種類の指定
flex-align	垂直方向の揃え
flex-flow	ボックス（flexbox）の方向
flex-line-pack	複数行のボックス（flexbox）の揃えの指定
flex-order	ボックス（flexbox）の表示順
flex-pack	水平方向の揃え

http://www.w3.org/TR/css3-flexbox/#property

04 ナビゲーションに「CSS Flexible Box Layout Module」の機能を適用



05 「Grid Layout」のプロパティ一覧（グリッドレイアウトの指定）

プロパティ	指定内容
grid-columns	列の幅の指定
grid-rows	行の高さの指定
grid-column	列の番号の指定
grid-row	行の番号の指定
grid-cell	グリッドセルの指定
grid-cell-stacking	グリッドセルの重なりの方向の指定
grid-template	グリッドセル構造の指定
grid-column-span	列の数の指定
grid-row-span	行の数の指定
grid-column-sizing	列の幅の算出方法の指定
grid-row-sizing	列の高さの算出方法の指定
grid-flow	方向の指定
grid-column-align	列の（水平方向）揃えの指定
grid-row-align	行の（垂直方向）揃えの指定
grid-layer	重ね順の指定

http://www.w3.org/TR/css3-multicol/#property-index

きます。続けて、行と列を指定し、分割された領域にコンテンツを配置していきます。

列は、grid-columnsプロパティで定義します。「grid-columns: 15% 60% 25%」の場合、1列目の幅が15%、2列目が60%、3列目が25%の幅になります。行は、grid-rowsプロパティで定義します。数値ではなく、「auto」を指定すると、内容量にあわせて自動的に値を算出してくれます。

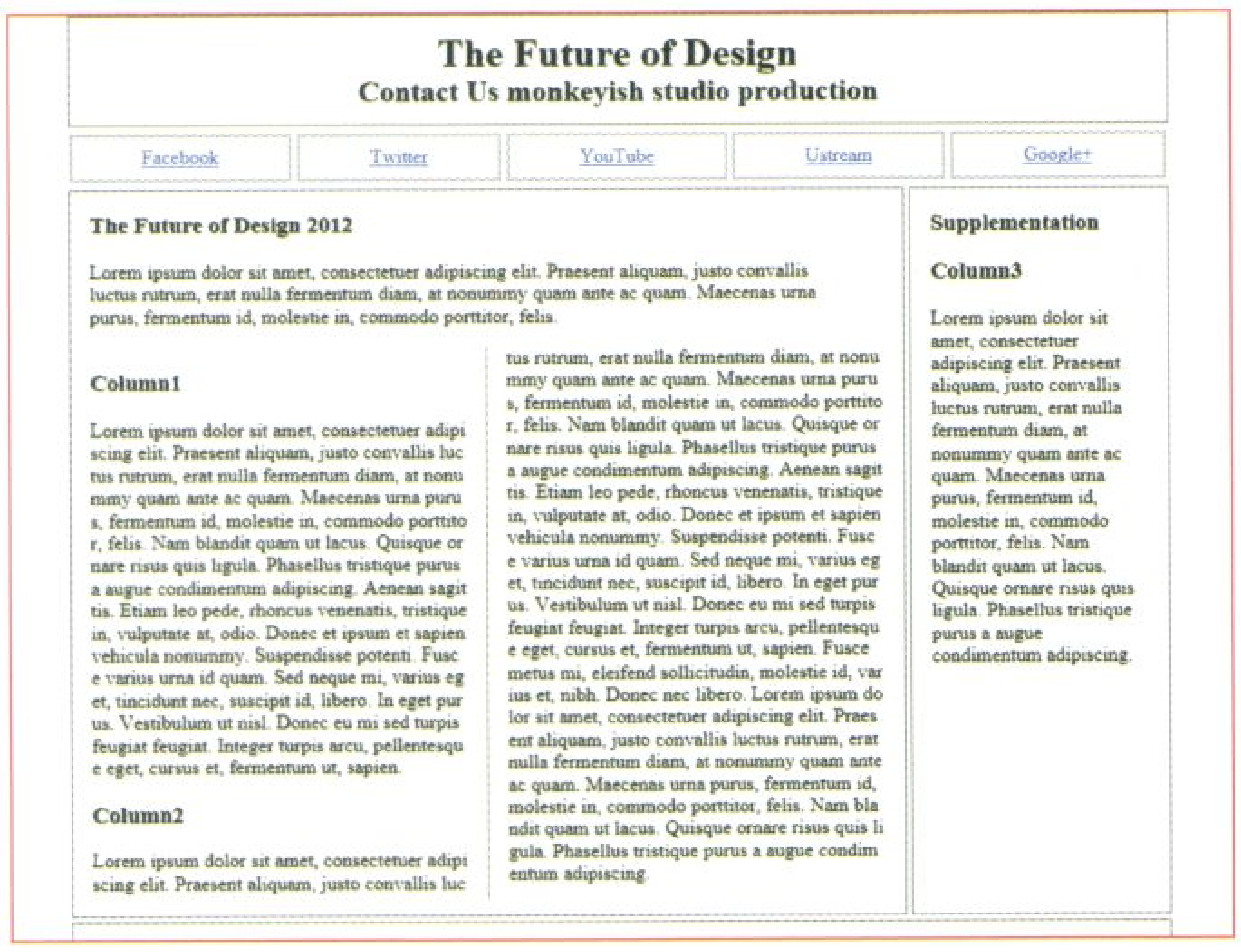
ボックスにdisplayプロパティを使用し、「grid」を指定すると、グリッドレイアウトが有効になります。列の横幅は、grid-columnsプロパティで数値もしくは比率（fr）を指定します（frは割合の単位です）05 06。

display

初期値	inline
適用要素	すべての要素
継承	なし
値	grid | inline-grid

```
#container {
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: 76% 24%;
  grid-columns: 76% 24%;
  width: 960px;
}
```

06 Webページの基本レイアウトに「Grid Layout」の機能を適用



その他のレイアウト仕様

「CSS Regions Module Level 3」07や「CSS Template Layout Module」08は、今まで不可能だった紙媒体の高度なページデザイン技術をブラウザ上で表現することができます。エディトリアルデザインの領域に近づくことで、電子書籍や電子雑誌などの基礎技術として普及する可能性もあります。

07 「CSS Regions Module Level 3」のプロパティ一覧（高度な流し込みレイアウトの指定）

プロパティ	指定内容
flow-into	指定されたフロー領域内に要素を置く
flow-from	指定されたフロー領域と結びつける
break-before	改ページ・改段の挿入位置（ボックスの前）の指定
fbreak-after	改ページ・改段の挿入位置（ボックスの後）の指定
break-inside	改ページ・改段の挿入位置（ボックス内）の指定
region-overflow	指定されたフロー領域と結びついた最後の領域の指定
@region	特定の領域に指定

http://www.w3.org/TR/css3-regions/

08 「CSS Template Layout Module」のプロパティ一覧（高度なポジショニングレイアウトの指定）

プロパティ	指定内容
display	要素が生成するボックスの種類の指定
position	ボックスの配置の指定
chains	スロットの連結の指定
break-before	改ページ・改段の挿入位置（ボックスの前）の指定
break-after	改ページ・改段の挿入位置（ボックスの後）の指定
break-inside	改ページ・改段の挿入位置（ボックス内）の指定

http://www.w3.org/TR/css3-layout/

メディアクエリ

CSS2.1には、メディアタイプを指定することができる「@media」があり、対象とするメディアに専用のスタイルを適用することができます。

メディアタイプには、screen（コンピュータの画面）、print（ページ出力）、handheld（小さな画面で狭帯域のハンドヘルドデバイス）、projection（プロジェクタなど）、speech（音声合成出力）など10種類があります。

CSS3では、「Media Queries」（メディアクエリ）が定義され、より詳細な条件でスタイルを適用できるようになりました 09 。デバイスのスクリーンサイズや解像度、縦横比、色数、デバイスの向き、ブラウザの横幅や高さを条件として、専用のスタイルを指定することができます。メディアクエリを使用することで、デバイスごとにコンテンツを最適化しなくても、柔軟に適応させることが可能になります 10 。

メディアクエリの基本的な記述例

```
<h1>タイトル</h1>

@media screen {
  h1 { color: red; }
}

@media all {
  h1 { color: red; }
}
```

メディア特性と条件を指定する記述例

```
@media screen and (min-width:960px) {
  h1 { color: red; }
}
```

ブラウザの表示領域の幅が960ピクセル以上の場合、タイトルの色を赤にする

```
@media screen and (min-width:480px) and (max-width:960px) {
  h1 { color: red; }
}
```

ブラウザの表示領域の幅が、480ピクセル以上、960ピクセル以下の場合、タイトルの色を赤にする

```
@media screen and (min-device-width:768px) and (max-device-width:1024px) {
  h1 { color: red; }
}
```

デバイスのスクリーンの幅が、768ピクセル以上、1024ピクセル以下の場合、タイトルの色を赤にする

デバイスの方向でスタイルを適用する記述例

```
@media screen and (orientation: landscape) {
  h1 { color: red; }
}
```

デバイスを横向き（ランドスケープモード）にした場合、タイトルの色を赤にする

09 Media Queries

http://www.w3.org/TR/css3-mediaqueries/

10 メディアクエリの主な特性、条件、値に関する一覧表

メディア特性	条件	値	最小/最大
width	ブラウザのウィンドウの幅	数値	min/max
height	ブラウザのウィンドウの高さ	数値	min/max
device-width	デバイスのスクリーンの幅	数値	min/max
device-height	デバイスのスクリーンの高さ	数値	min/max
aspect-ratio	ブラウザのウィンドウの縦横比	比率(横/縦)	min/max
device-aspect-ratio	デバイスのスクリーンの縦横比	比率(横/縦)	min/max
orientation	デバイスの方向	portrait(縦)/landscape(横)	-
resolution	解像度	dpi, dpcm	min/max
color	色数(ビット数)	数値	min/max
color-index	カラーテーブル	数値	min/max
monochrome	グレースケールの諧調(ビット数)	数値	min/max

CHAPTER 01

レイアウト

Layout

マルチデバイスに対応したレスポンシブなWebページ
flexboxを使った柔軟なボックスレイアウト
ナビゲーションの位置を固定したWebページ
ウィンドウサイズに連動するリキッドタイプのフリーレイアウト
グリッドレイアウトで実現するレスポンシブなサイト
透過画像を効果的に組み合わせたデザイン
CSS3と画像を組み合わせたアナログテイストのデザイン
テキストが読みやすい段組みレイアウト

01 マルチデバイスに対応した レスポンシブなWebページ



1つのソースで、PC、タブレット、スマートフォンなど、画面サイズの異なる複数のデバイスに対応するレスポンシブWebデザインを採用したWebページを紹介します。メディアクエリを利用して、各デバイスのディスプレイの幅に応じた表示に切り替えます。

制作・文／山本圭助
（株式会社ラナデザインアソシエイツ）

制作のポイント

- レスポンシブWebデザインの考え方
- メディアクエリの使い方
- Internet Explorerへの対応

対応ブラウザ

IE	P.E.
Safari	5.1 以上
Firefox	8 以上
Chrome	15 以上

※IE8以下はコンディショナルコメントで対応。

TECHNIC 1

ワンソースで多くのデバイスに対応

レスポンシブWebデザインとは、Webサイトを閲覧する環境にあわせて最適なレイアウトを表示させる方法で、1つのソースで多くのデバイスに対応させることができます。



TECHNIC 2

メディアクエリの使用

画面サイズに合わせてスタイルを切り替えるには、メディアクエリを使用します。





メディアクエリの使い方

01

メディアクエリとは、link要素とstyle要素のmedia属性で、表示対象となるメディアの種類を特定することを可能にします 01-1 。@importでも使用できます。
媒体型と媒体特性を、キーワードを使用して指定します 01-2 01-3 。

MEMO

サンプルではコンテンツをタブの形式でレイアウトしています。この手法についてはCHAPTER02-02「表示内容を切り替えるタブナビゲーションの実装」（96ページ）で詳しく解説しています。

01-1 メディアクエリ

要素	記述
キーワード	カンマ（OR検索）、and、not
主な媒体型	screen（一般的なディスプレイ）、projection（プロジェクター）、print（プリンタ）、tv（テレビ）、all（すべて）など
主な媒体特性	width、device-width、height、max-height、aspect-ratio、orientation、など

01-2 画面幅360px以下の指定（link要素の場合）

```
<link rel="stylesheet" href="css/style.css" media="screen and (max-width: 360px)">
```

01-3 画面幅360px以下の指定（style要素内の場合）

```
<style>
@media screen and (max-width: 360px) {
— 省略 —
}
</style>
```

02

画面幅が921px以上、920px～361px、360px以下の3種類を表示分けします 02-1 02-2 02-3 。

02-1 画面幅921px以上



02-2 画面幅920px～361px



02-3 画面幅360px以下



03

ベースになるHTMLを用意します03-1。

03-1 ベースになるHTML

```
<body>
  // ヘッダー
  <div id="header">
    <img src="img/logo.png" class="logo">
    <ul class="nav">
      <li>About</li><li>Works</li><li>Access</li><li>Contact</li>
    </ul>
  </div>
  // メインビジュアル
  <div id="area-visual">
    <div class="container">
      <div class="content">
        <h1>we create stylish design.</h1>
        <p>Lorem Ipsum ...</p>
      </div>
    </div>
  </div>
  — 省略 —
</body>
```

04

次に、レイアウトによってサイズ変更などがない共通のスタイル指定を記述します04-1。

04-1 サイズ変更に影響を受けない共通のスタイルを指定

```
// ヘッダーエリアを中央揃えに。
#header { margin: 0 auto; }
// ロゴとナビゲーションはfloatで横並びにします。
#header .logo { float: left; }
#header .nav { float: right; }

// メインビジュアルエリアも中央揃えに。
// ビジュアルは背景画像とし、右揃えに。
#area-visual .container {
  margin: 0 auto;
  background: url(../img/visual.jpg)
  no-repeat right top #ffe24d;
}
```

画面サイズに応じた画面幅の設定

05

画面サイズごとのスタイル設定を加えていきます。
画面幅が921px以上（＝最小画面幅が921px）の場合、min-widthで最小画面幅を指定します05-1 05-2。

05-1 min-widthで最小画面幅を指定

```
@media screen and (min-width: 921px) {
  #header {
    width: 888px;
    height: 60px;
  }
  #area-visual .container {
    width: 888px;
    height: 276px;
  }
  — 省略 —
}
```

05-2 画面幅が広いときのヘッダーとメインビジュアル





06

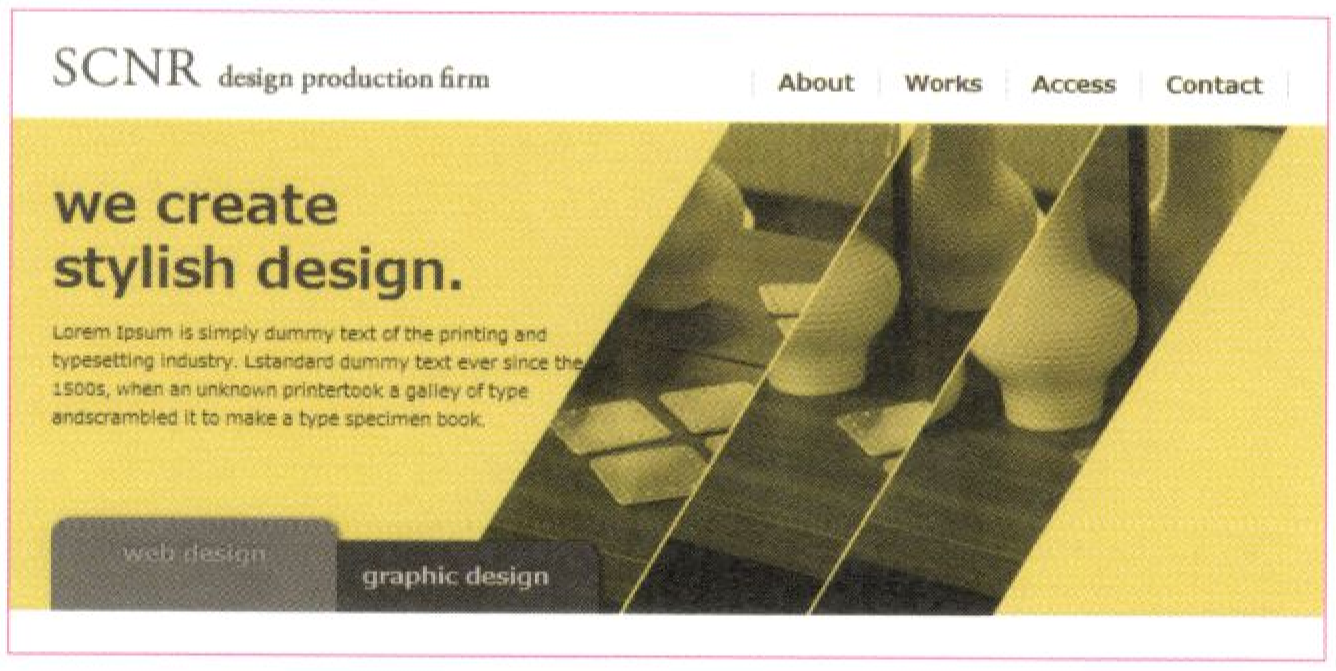
画面幅が920px〜361px（＝最大画面幅が920pxで最小画面幅が360px）の場合、min-widthで最小画面幅を、max-widthで最大画面幅を指定します 06-1 06-2。

06-1 min-widthで最小画面幅を、max-widthで最大画面幅を指定

```
@media screen and (min-width: 361px) and (max-width: 920px) {
  #header {
    width: 674px;
    padding: 0 24px;
  }

  #area-visual .container {
    width: 674px;
    height: 276px;
  }
  ― 省略 ―
}
```

06-2 画面幅が中ぐらいのときのヘッダーとメインビジュアル



07

画面幅が360px以下（＝最大画面幅が360px）の場合、max-widthで最大画面幅を指定します 07-1 07-2。
これで、画面幅によってレイアウトが自動的に変更されるようになりました。

07-1 max-widthで最大画面幅を指定

```
@media screen and (max-width: 360px) {
  #header {
    width: 336px;
    padding: 0 12px;
  }
  // ナビゲーションがロゴの下にくるように
  変更します。
  #header .nav {
    float: left;
  }

  #area-visual {
    height: 204px;
  }
  #area-visual .container {
    padding: 0 12px;
    width: 336px;
    height: 204px;
  }
  ― 省略 ―
}
```

07-2 画面幅が小さいときのヘッダーとメインビジュアル



Internet Explorerへの対応

08

閲覧環境を特定できるメディアクエリですが、IE8以下ではメディアタイプまでしか対応していません。
そこでIEの場合には、IE用のスタイルシートを作成し、コンディショナルコメントで読み込み対応します。

MEMO

メディアタイプはメディアクエリの前身で、記述方法は同じですが、指定できるキーワードは媒体型だけです。カンマ区切りで複数指定できます 01。

01 デバイスがプリンタまたはテレビの場合のスタイル指定

```
<link rel="stylesheet" href="style.css" media="print, tv">
```

*COLUMN

コンディショナルコメント

コンディショナルコメントとは、htmlファイルのコメント記述におけるIEの独自拡張で、次のように指定した条件に合ったブラウザのみ、処理を実行することができます 01 02 03。

01 コンディショナルコメントの書式

```
<!--[if IEのバージョン]>
  処理したい内容
<![endif]-->
```

02 IEのバージョン指定方法

表記	条件
gt	より大きい(greater than)
lt	未満(less than)
gte	以上(greater than equal)
lte	以下(less than equal)

03 バージョン指定の例

記述	対応
IE	全バージョン
IE 6	IE6のみ（6.0という指定も可）
lt IE 7	IE7未満（IE6以下）
gte IE 8	IE8以上

09

コンディショナルコメントのみで対応させる場合は、IE8以下のバージョンでは固定レイアウトにします。過去のIEで閲覧されているということは、スマートフォンでの閲覧を想定する必要はありませんので、実際の運用上で問題になることはないはずです。
まず、ここまで記述した画面幅ごとのCSSコードをファイル単位に分割します（921px以上：large.css、920px～361px：middle.css、360px以　下：small.css） 09-1。
HTMLではコンディショナルコメントを利用し、IEのバージョンに応じてlink要素で読み込むCSSを変更します 09-2。IE7、8ではlarge.css、IE6ではmiddle.cssを読み込むことで、過去のIEでは固定レイアウトのページとして表示されます。

09-1 large.css（middle.css、small.cssも同様）

```
@charset "UTF-8";
#header {
 width: 888px;
 height: 60px;
}

#area-visual .container {
 width: 888px;
 height: 276px;
}
#area-visual .container .content {
 width: 47.297%;
}

#area-content .container {
 width: 888px;
}
#area-content .container .content {
 padding: 12px 0 12px 12px;
}
```

09-2 画面幅に応じた設定ファイルを読み込む

```
<!--[if lte IE 8]>
<!--[if gte IE 7]>
  <link rel="stylesheet" href="css/large.css">
<![endif]-->
<![endif]-->

<!--[if IE 6]>
  <link rel="stylesheet" href="css/middle.css">
<![endif]-->
```



MEMO

IE9ではメディアクエリに対応しているので、サンプルは問題なく動作します。

TIPS

ここでIE7～8ではlarge.css、IE6ではmiddle.cssを読み込んでいます。つまり、IE6では920px～361pxのレイアウトで固定しているわけですが、これは古いバージョンであるIE6で閲覧されている環境では、ディスプレイが小さいケースが多いことが想定されるためです。

10

また、IE8以下でもメディアクエリを利用できるようにする「css3-mediaqueries.js」10-1 というJavaScriptライブラリがあるので、それを使用して対応させることもできます。この場合は、HTMLのhead要素部分を 10-2 のようにすれば利用できます。

ただし、css3-mediaqueries.jsはlink要素内のmedia属性を利用したメディアクエリには対応していません。10-3 のような場合はスタイルが正しく適用されないので注意しましょう。

10-1 css3-mediaqueries.js

http://code.google.com/p/css3-mediaqueries-js/

10-2 IEのバージョンに応じた設定ファイルを読み込む

```
<head>
  <link rel="stylesheet" href="css/style.css"> // 全サイズ版
  <!—-[if lt IE 9]><script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/
  css3-mediaqueries.js"></script><![endif]—->
</head>
```

10-3 スタイルが正しく適用されない例

```
// HTML
<head>
  <link rel="stylesheet" href="css/style.css"> // 共通用
  <link rel="stylesheet" href="css/large.css" media="screen and (min-width: 921px)">
  <link rel="stylesheet" href="css/middle.css" media="screen and (min-width: 361px) and
  (max-width: 920px)">
  <link rel="stylesheet" href="css/small.css" media="screen and (max-width: 360px)">
  <!—-[if lt IE 9]><script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/
  css3-mediaqueries.js"></script><![endif]—->
</head>
```

MEMO

JavaScriptライブラリを使用して対応させる場合は、ローカル環境では動作しない場合がある点、表示速度が遅くなる可能性がある点に注意しましょう。基本的には過去のIEのバージョンで閲覧されている環境はPCに限定できることから、無理にレスポンシブレイアウトに対応させる必要はそれほどありません。09 のように固定レイアウトにして対応させたほうが軽快に閲覧できるなど、ユーザーにとってもメリットが大きい場合もあります。

まとめ [CONCLUSION]

メディアクエリを使うことによって、1つのソースで多用な閲覧環境に対応することが可能です。
しかし、その反面データを詰め込み過ぎてしまうことにもつながります。

PCやスマートフォンで表現したい内容の差が小さい場合は、メディアクエリを利用する、差が大きい場合はそれぞれ別々のソースで対応するなど、それぞれのケースで使い分ける工夫をしてみてください。

02 flexboxを使った柔軟なボックスレイアウト



3カラムのリキッドレイアウトを実装するには、これまではdiv要素を入れ子にし、floatとネガティブマージンを組み合わせて実装する必要がありました。しかし、flexboxを使えば、floatよりも柔軟に対応できるようになります。

制作・文／山田敬美

制作のポイント

- floatレイアウトとの違いを理解する
- リキッドレイアウトや表示順の変更をプロパティひとつで簡単に実装
- box-sizingなど、ほかのCSS3プロパティとの組み合わせ

対応ブラウザ

IE	10 以上	Safari	3.2 以上
Firefox	3 以上	Chrome	5 以上

TECHNIC 1

旧仕様と現仕様の両方を指定

flexboxは2011年3月に仕様が変更になったため、旧仕様と現仕様の両方を指定しています。

```
#contents {
    /* Old */
    display: -webkit-box;
    display: -moz-box;
    display: -ms-box;
    display: box;
    /* New */
    display: -webkit-flexbox;
    display: -moz-flexbox;
    display: -ms-flexbox;
    display: flexbox;
}
```

TECHNIC 2

画面サイズに応じたレイアウトに

メディアクエリとの組み合わせで、画面サイズに応じたレイアウトにすることができます。



flexboxの概要

01

カラムをレイアウトするために、floatプロパティを使う方法は一般的ですが、floatした要素の高さが揃わなかったり、floatをわざわざクリアする必要があるなど、不便に感じることが多いのではないでしょうか。その原因は、floatはもともと段組みをするためのプロパティではないからです。floatの代わりに、CSS3で導入されたflexboxを使えば、これまでの悩みが嘘のように、簡単にカラムをレイアウトできるようになります01-1。

01-1 floatレイアウトとflexboxの違い

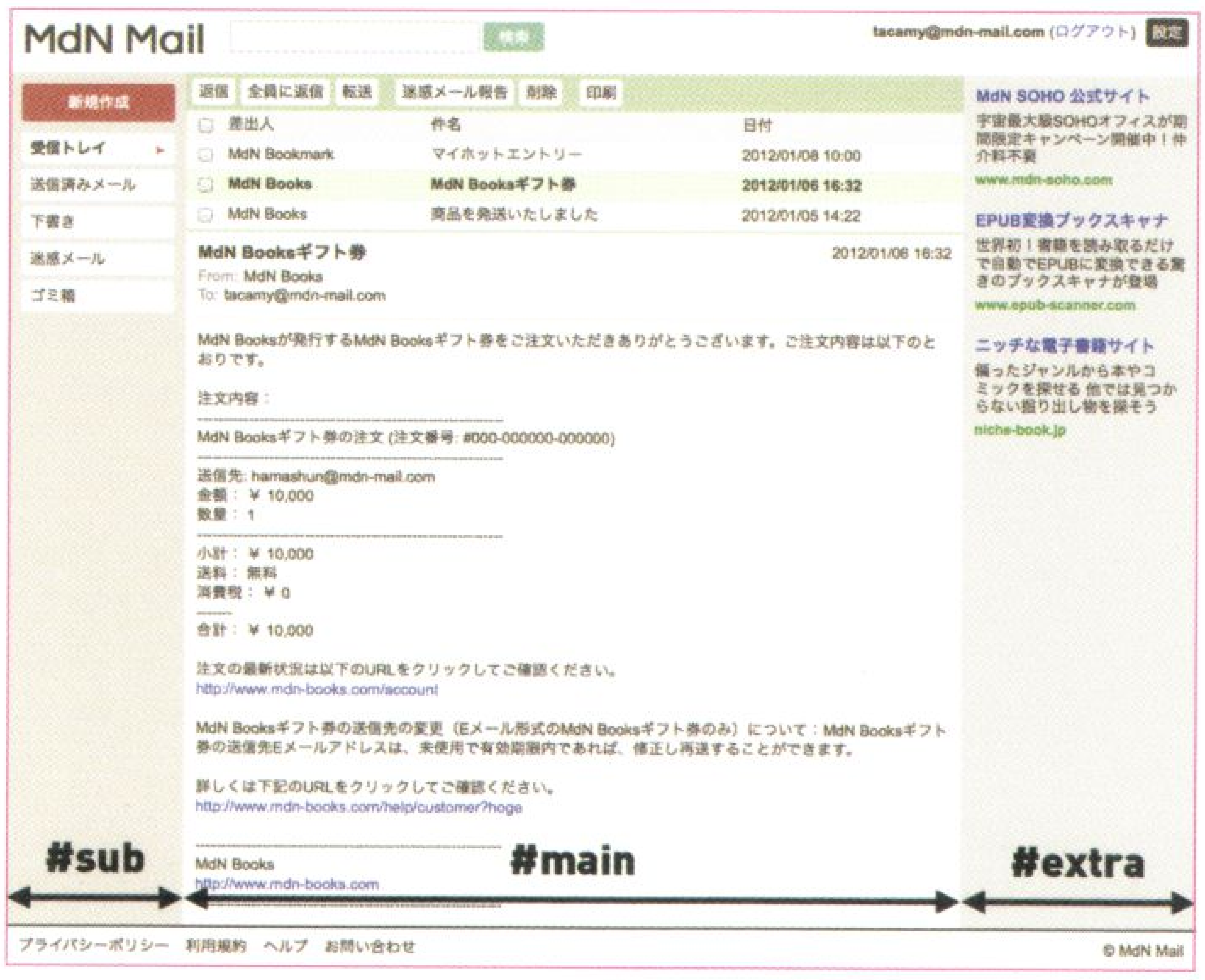
比較内容	float	flexbox
コンテナとなる親要素の高さ	子要素の高さは無視される(Clearfixでの対応は可能)	子要素の高さが含まれる
子要素の高さが異なる場合	それぞれのボックスが独立した高さを持ち、それらを揃えることはできない	一番高いボックスに合わせてすべてのボックスの高さが揃う
後に続く要素への影響	floatをクリアしないと、後ろの要素に影響する	影響なし
要素の順番を変更	ある程度は可能だが制限がある	自由に変更できる
固定幅+リキッドなレイアウトを組む	div要素の複雑な入れ子とネガティブマージンを駆使すれば可能	プロパティを1行書くだけで対応可能

ベースとなるHTMLとCSSの記述

02

左右のカラムが固定幅で、中央のカラムがリキッドという3カラムのレイアウトを、flexboxを使って実装していきます02-1。

02-1 完成例



03

ベースとなるHTMLは03-1のようにシンプルに記述します。順番は後から自由に変えられるため、メインカラムを先頭に記述します。CSSでは、サブカラムとエキストラカラムの幅と、各カラムの背景色を指定します03-2。この時点では、それぞれのカラムは縦に並んでいます03-3。

03-1 ベースのHTML

```
<div id="contents">
    <div id="main"><!--メインカラム--></div>
    <div id="sub"><!--サブカラム--></div>
    <div id="extra"><!--エキストラカラム--></div>
</div>
```

03-2 ベースのCSS

```
#contents {
    width: 100%;
}
#main {
    background: #fff;
}
#sub {
    width: 150px;
    padding: 10px;
    background: #edebe6;
}
#extra {
    width: 200px;
    padding: 10px;
    background: #eee;
}
```

03-3 この時点では、各カラムが縦に並んでいる



flexboxで3カラムレイアウトにする

04

各カラムを横に並べたレイアウトにするには、コンテナとなる親要素に対して、display: flexbox;（旧仕様ではdisplay: box;）を指定するだけです **04-1**。親要素の直下にある兄弟要素が横に並び、それぞれのカラムの高さが揃います **04-2**。

ATTENTION!

最新の仕様書ではdisplay: flexbox;と記載されていますが、現行の主要ブラウザはどれもまだ未実装です。そのため、古い仕様であるdisplay: box;も併せて指定します。なお、新しい仕様の動作確認を行いたい場合は、現時点ではChrome Canary（正式版のChromeに導入されていない開発中の機能がいち早く導入される）で実装されているため、そちらで確認ができます。

MEMO

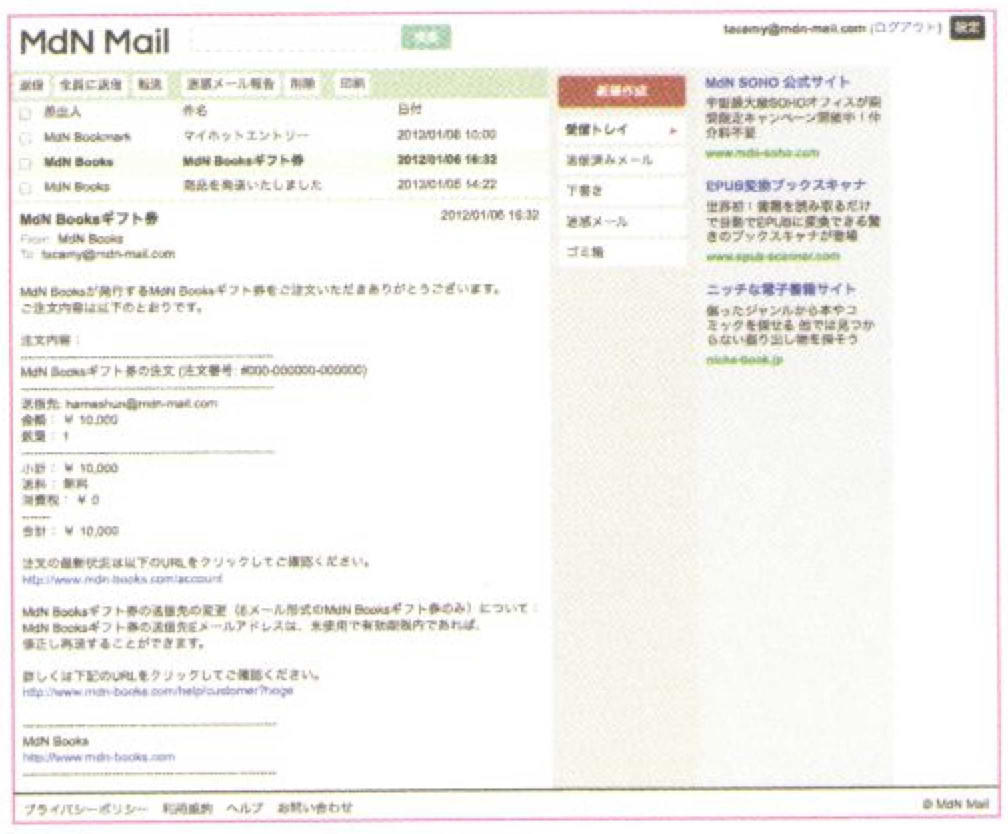
IEでもIE10 Platform Previewからflexboxに対応しているため、「-ms-」のベンダープレフィックスもつける必要があります（ベンダープレフィックスについては、15ページのINTRODUCTION「ブラウザとベンダープレフィックス」の項を参照）。

04-1 親要素にdisplay: flexbox;を指定してカラムを横並びにする

```
#contents {
    /* Old */
    display: -webkit-box;
    display: -moz-box;
    display: -ms-box;
    display: box;
    /* New */
    display: -webkit-flexbox;
    display: -moz-flexbox;
    display: -ms-flexbox;
    display: flexbox;

    width: 100%;
}
```

04-2 カラムが横並びになり、高さが揃った状態となった





05

親要素の幅に合わせて、メインカラムの横幅が変わるようにします。flex()ファンクション（旧仕様ではbox-flexプロパティ）を使い、()の中に入る数字を「1」と指定することで、親要素に合わせて幅を伸縮させることができます 05-1 05-2。

MEMO

flex()ファンクションは、その他のflexboxプロパティとは異なり、width、height、margin、paddingプロパティと組み合わせて使います。

05-1 メインカラムをリキッドにする指定

```
#main {
    /* Old */
    -webkit-box-flex: 1;
    -moz-box-flex: 1;
    -ms-box-flex: 1;
    box-flex: 1;
    /* New */
    width: -webkit-flex(1);
    width: -moz-flex(1);
    width: -ms-flex(1);
    width: flex(1);

    background: #fff;
}
```

05-2 メインカラムの幅が、親要素に合わせて伸縮するようになった



06

左から2番目にあるサブカラムを、1番目に変更します。カラムの表示順を変更するには、flex-orderプロパティ（旧仕様ではbox-ordinal-group）を使います。プロパティの値にはそれぞれ、表示させたい順番を指定します。ここでは、1：サブカラム、2：メインカラム、3：エキストラカラムの順で指定しています 06-1 06-2。

06-1 表示順を「サブ→メイン→エキストラ」の順に変更



06-2 flex-order: N; で表示順を指定

```
#main {
    /* Old */
    -webkit-box-ordinal-group: 2;
    -moz-box-ordinal-group: 2;
    -ms-box-ordinal-group: 2;
    box-ordinal-group: 2;
    /* New */
    -webkit-flex-order: 2;
    -moz-flex-order: 2;
    -ms-flex-order: 2;
    flex-order: 2;

    /* Old */
    -webkit-box-flex: 1;
    -moz-box-flex: 1;
    -ms-box-flex: 1;
    box-flex: 1;
    /* New */
    width: -webkit-flex(1);
    width: -moz-flex(1);
    width: -ms-flex(1);
    width: flex(1);

    background: #fff;
}
#sub {
    /* Old */
    -webkit-box-ordinal-group: 1;
    -moz-box-ordinal-group: 1;
    -ms-box-ordinal-group: 1;
    box-ordinal-group: 1;
    /* New */
    -webkit-flex-order: 1;
    -moz-flex-order: 1;
    -ms-flex-order: 1;
    flex-order: 1;

    width: 150px;
    padding: 10px;
    background: #edebe6;
}
#extra {
    /* Old */
    -webkit-box-ordinal-group: 3;
    -moz-box-ordinal-group: 3;
    -ms-box-ordinal-group: 3;
    box-ordinal-group: 3;
    /* New */
    -webkit-flex-order: 3;
    -moz-flex-order: 3;
    -ms-flex-order: 3;
    flex-order: 3;

    width: 200px;
    padding: 10px;
    background: #eee;
}
```

07

現時点では、WebKit系ブラウザ（SafariやChromeなど）とFirefoxで、flexboxのpaddingをwidthに含めるかどうかの挙動が異なります。そのため、カラムにwidthとpaddingを両方指定すると、ブラウザごとにカラムの幅が異なります。この差異をなくすために、CSS3のbox-sizingプロパティの値をborder-boxに指定して、カラムの幅にpaddingを含めるよう調整します 07-1 07-2。

ATTENTION!

flexboxに未対応のIEでもflexboxを使えるようにするJavaScriptライブラリを使う場合、box-sizingを使っていると、IE8で不要な隙間が空いてしまう場合があります。このような場合には、box-sizingは使わずに、widthとpaddingをそれぞれ別々の要素に指定するようにします。

07-1 widthとpaddingが指定されているカラムに対して、box-sizing: border-box;を指定

```
#sub {
    /* Old */
    -webkit-box-ordinal-group: 1;
    -moz-box-ordinal-group: 1;
    -ms-box-ordinal-group: 1;
    box-ordinal-group: 1;
    /* New */
    -webkit-flex-order: 1;
    -moz-flex-order: 1;
    -ms-flex-order: 1;
    flex-order: 1;

    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    -ms-box-sizing: border-box;
    box-sizing: border-box;

    width: 150px;
    padding: 10px;
    background: #edebe6;
}
```

```
#extra {
    /* Old */
    -webkit-box-ordinal-group: 3;
    -moz-box-ordinal-group: 3;
    -ms-box-ordinal-group: 3;
    box-ordinal-group: 3;
    /* New */
    -webkit-flex-order: 3;
    -moz-flex-order: 3;
    -ms-flex-order: 3;
    flex-order: 3;

    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    -ms-box-sizing: border-box;
    box-sizing: border-box;

    width: 200px;
    padding: 10px;
    background: #eee;
}
```



07-2 Chrome(上)とFirefox(下)でサブカラムとエキストラカラムの幅がpaddingの分だけ異なる



→

box-sizing適用後、Chrome(上)とFirefox(下)で同じ幅になった



画面の幅に合わせて3カラムから2カラムに切り替える

08

次に、画面サイズに応じてレイアウトが変わるようにして、どの画面サイズでも見やすくなるように調整します。例として、ブラウザサイズが800px以下の場合は、広告のあるエキストラカラムを非表示にして、メインカラムを広くとれるようにします。**08-1**はエキストラカラムが表示され、**08-2**は非表示になっています。

08-1 画面サイズが1024pxの場合の表示



08-2 画面サイズが800pxの場合の表示



09

画面サイズによって異なるスタイルを指定するには、CSS3のメディアクエリを使います。メディアクエリにはいくつかの指定方法がありますが、今回はCSSファイルに記述する方法を採用します。@media screen and (max-width: 800px)とすることで、「画面サイズが800px以下の場合に処理をする」という意味になります09-1。

09-1 画面サイズが800px以下の場合には、エキストラカラムを非表示にする

```
@media screen and (max-width: 800px) {
    #extra {
        display: none;
    }
}
```

ライブラリを使用してIEに対応する

10

「display: flexbox;」は現状どの主要ブラウザも実装していませんが、旧仕様のdisplay: box;であれば、Chrome、Safari、Firefoxですでに実装されています。Internet ExplorerではIE10 Platform Previewから実装されていますが、IE6からIE9で実装するには、JavaScriptを使って対応する必要があります10-1。そのため、今回は「flexie.js」というライブラリを使用します。

ATTENTION!

flexie.jsを読み込む前に、jQueryを読み込んでおく必要があります。

10-1 デフォルト状態でのIE9の表示





11

使い方は、「Flexie」のWebサイト 11-1 からflexie.js（バージョン1.0.2）をダウンロードし、HTMLのhead内で読み込むだけです 11-2 。ただし、07 でbox-sizingを指定していた場合、11-3 のように下部に隙間が空く場合があります。IE8では同様に右側と下部に隙間が空き、IE7とIE6は問題ありませんでした。flexie.jsを使用する場合は、カラムに横幅を指定したら、paddingは別の要素に指定するようにして、box-sizingは使わない方が無難でしょう。

ATTENTION!

IEではflexie.jsの最新バージョン（1.0.3）は動かなかったため、1つ前のバージョン（1.0.2）を使用します。
また、flexie.jsは、ローカル環境やDropboxのPublicなどでは動作しません。Apache等でローカルにサーバーを立てるか、実際にサーバーにアップロードして動作を確認する必要があります。

MEMO

IE以外のブラウザやIE10以上では必要ないため、条件分岐コメントを使ってIE6からIE9にのみ読み込まれるよう指定します。

11-1 「Flexie」のWebサイト（flexie.js）

http://flexiejs.com/

11-2 head内でjQueryとflexie.jsを読み込む

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
<!--[if (gte IE 6)&(lte IE 9)]><script src="/js/lib/flexie.js"></script><![endif]-->
```

11-3 flexie.jsを適用したIE9の表示



まとめ [CONCLUSION]

flexie.jsはflexboxに未対応のIEとOpera、両方に対応しているように記載されていますが、今回のサンプルでは、IEではある程度レイアウトが再現できましたが、Operaでは動きませんでした（そのため、条件分岐コメントでIE6からIE9でのみ読み込んでいます）。また、IEの場合でも、flexie.jsの最新バージョン（1.0.3）では動かなかったため、1つ前のバージョンで動作させています。このように、まだ不安定な部分もあるようです。そもそも、JavaScriptがオフの環境では完全にレイアウトが崩壊することになるため、これらのことを踏まえると、PC向けの大きな案件で利用するにはまだリスクがあります。
ただし、タブレットやスマートフォン限定、またはPC向けでもIEやOperaは対象外という条件の場合には、flexboxは本当に便利なプロパティです。floatでは実現できないような複雑なレイアウトも簡単に実装できるので、案件の条件によっては今からでも積極的に使っていけるでしょう。

03 ナビゲーションの位置を固定したWebページ



メニューナビゲーションの位置が固定されたページを作成します。ページを下にスクロールしてもヘッダーメニューの位置が変わらない、使いやすいデザインになっています。transformプロパティなどを利用して、メニューのロールオーバー時に動きをつけました。

制作・文／藤本勝己（株式会社レリーフ）

制作のポイント

- position: fixedでヘッダーを固定する
- CSSだけでロールオーバーのスムーズな動きを実装
- transitionプロパティの設定

対応ブラウザ

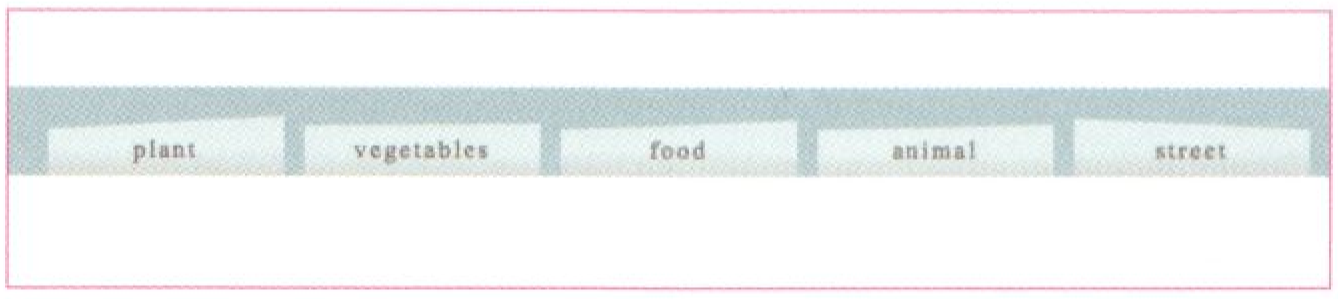
IE	P.E.	Safari	5.1 以上
Firefox	9 以上	Chrome	16 以上

※IEはtransitionに非対応。

TECHNIC 1

ヘッダーの高さはコンパクトなサイズに設計

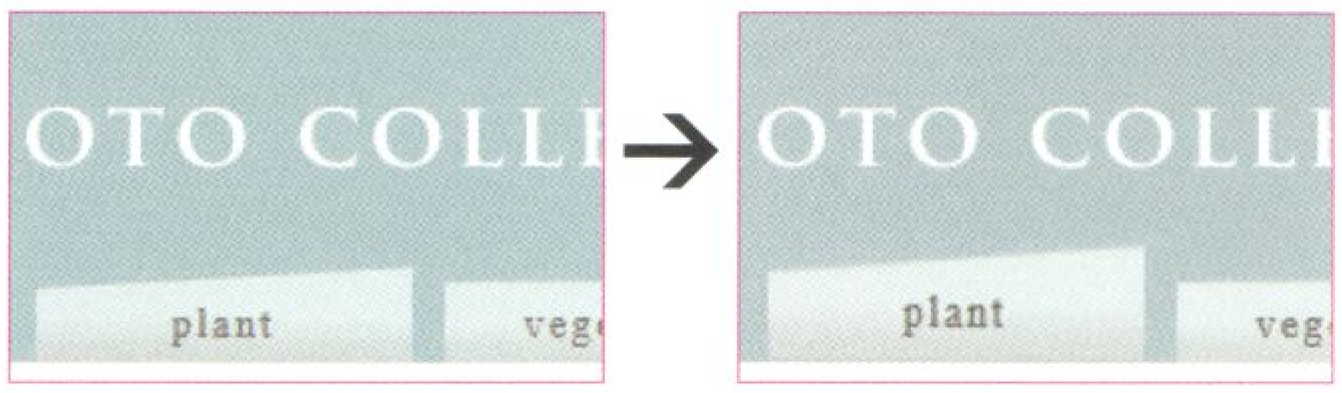
ヘッダーを固定で表示した場合に、ヘッダーの高さ（height）がありすぎると、ウィンドウのサイズによってはコンテンツの表示幅が狭まり見づらくなるため、細めの設計にしています。



TECHNIC 2

メニューにロールオーバー時の動きをつける

transformプロパティとtransitionプロパティを併用して、メニューをロールオーバーした際の動きを表現しています。





ヘッダーとフッターを固定する

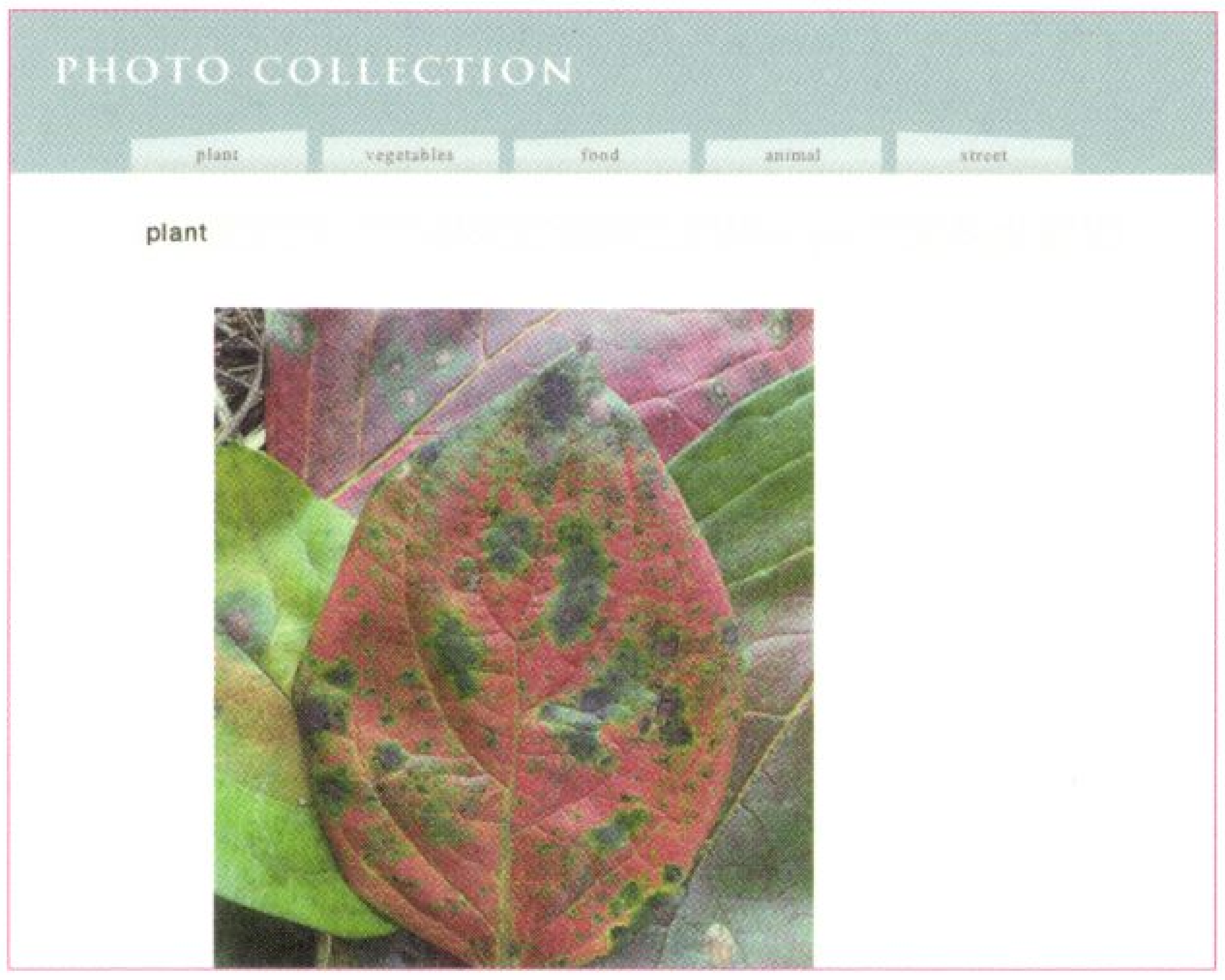
01

position: fixed; を使うと、フレームセットのように画面の一部を固定して、それ以外の部分のみスクロールするように作ることができます 01-1。

MEMO

position: fixed; はIE6では対応していないため、ヘッダーメニューとフッターの位置を固定することができません。

01-1 ヘッダー部分を固定する



02

今回のレイアウトで指定しているIDは、02-1 のようになります。
固定したいheader部分にposition:fixedを指定すると、ヘッダーを固定することができます。ここではフッターも同じ要領で固定しています。
headerに設定するCSSは、02-2 のようになります。

02-1 IDの指定



02-2 headerに指定するCSS

```
#header {
  background-color: #A0C9D8;
  color: #fff;
  position: fixed !important;
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  width: 100%;
  height: 30px;
  font-size: 12px;
  padding-top: 45px;
  padding-left: 30px;
  font-family: "Times New Roman", Times, serif;
— 省略 —
}
```

transform:rotateでロールオーバーを設定

03

transformプロパティを使って、ヘッダーメニューにロールオーバーした時の動きをつける設定を行います。

transformプロパティには、noneかtransform関数を指定することができます。

transform関数を指定すると、生成されたボックスを変形・移動・回転させることができます 03-1。noneは変形しない、つまりtransformを無効にするという値です。

03-1 transform関数

値	効果
rotate(deg)	回転（degで指定。10degや-10degなど）
scale(xy[x,y])	拡大・縮小（数値を指定。1や0.5など）
skew(deg)	X軸、Y軸へのゆがみ（degで指定。10degや-10degなど）
translate(xy[x,y])	移動（pxで指定。10pxなど）

04

#headerのナビゲーションを設定しているli:hoverの情報に、transformを設定します。各ブラウザに対応するための-webkit-、-moz-、-ms-、-o-も同じく設定します 04-1 04-2 04-3。

04-1 ヘッダー部分

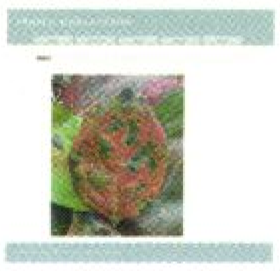


04-2 li:hoverにtransformと、各ブラウザへの対応を指定する

```
#header ul li:hover {
  -webkit-transform:scaleY(1.2) translateY(-3px);
  -moz-transform:scaleY(1.2) translateY(-3px);
  -ms-transform:scaleY(1.2) translateY(-3px);
  -o-transform:scaleY(1.2) translateY(-3px);
  transform:scaleY(1.2) translateY(-3px);
— 省略 —
}
```

04-3 #headerのナビゲーションを設定

```
<div id="header">
 <ul>
  <li id="nav01"><a href="#plant">plant</a></li>
  <li id="nav02"><a href="#vegetables">vegetables</a></li>
  <li id="nav03"><a href="#food">food</a></li>
  <li id="nav04"><a href="#animal"> animal</a></li>
  <li id="nav05"><a href="#street">street</a></li>
 </ul>
</div>
```



transitionで変化を設定

05

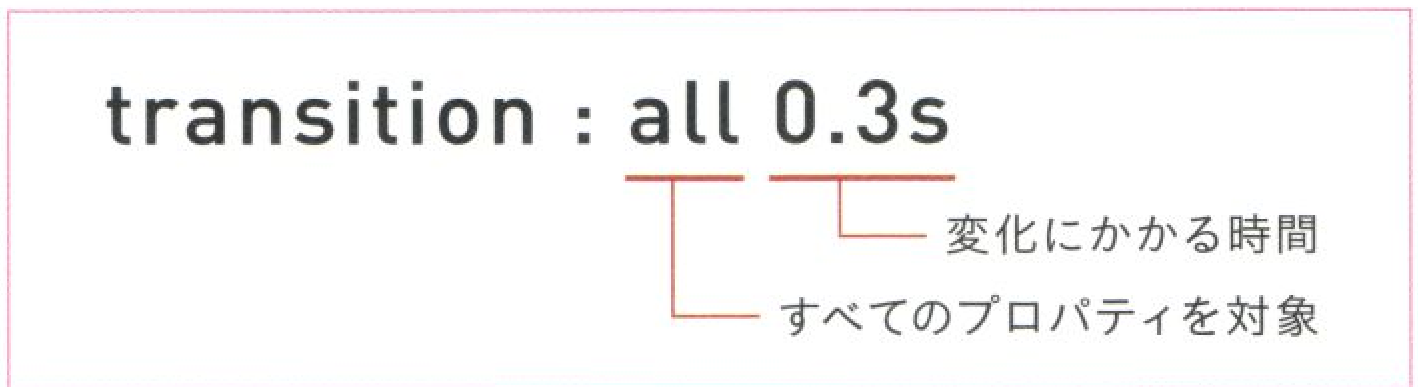
transitionプロパティでは、transition-property（変化させる対象のCSSプロパティ名を指定）、transition-duration（値が変化するのに掛かる時間を指定）、transition-delay（変化が始まるまでの時間を指定）などを、まとめて指定することができます 05-1。

各値は半角スペースで区切って指定します。時間を設定する値のうち最初に指定された値がtransition-duration、2番目に設定された値がtransition-delayの値になります。ここではtransition-durationだけを指定し、transition-delayは指定していません。transition-delayの初期値は「0」のため、transitionによる変化がすぐに開始する設定になっています。

MEMO

transitionプロパティについてはCHAPTER02-02「表示内容を切り替えるタブナビゲーションの実装」、100ページのCOLUMNもご覧ください。

05-1 transitionプロパティの基本的な記述



06

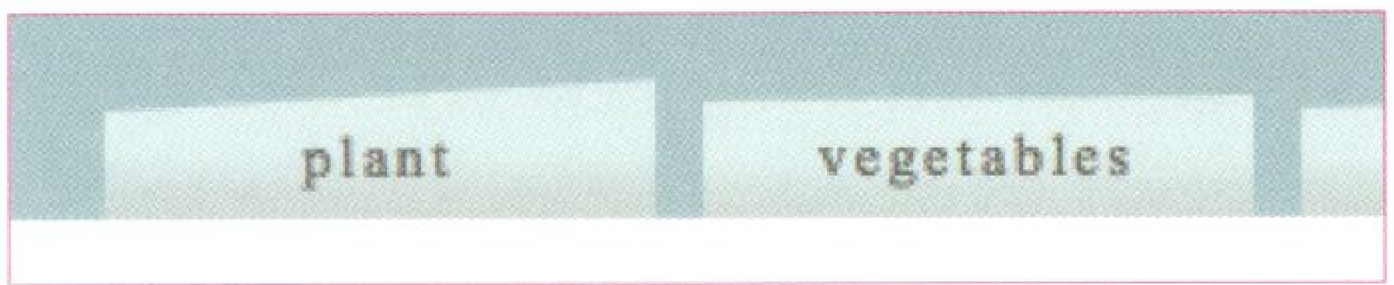
今回のサンプルでは、マウスオーバーした際に、少しだけ上に動く設定にしています 06-1。

今までFlashやgifアニメーションで作っていたような動きを実現できます 06-2 06-3。

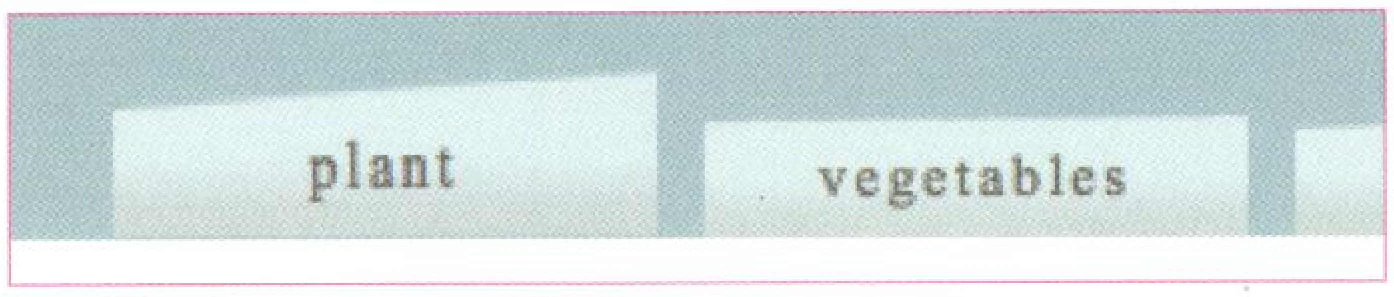
06-1 transitionの時間を指定する

```
#header ul li:hover {
― 省略 ―
  -webkit-transition:all 0.3s;
  -moz-transition: all 0.3s;
  -o-transition: all 0.3s;
  transition: all 0.3s;
}
```

06-2 通常時



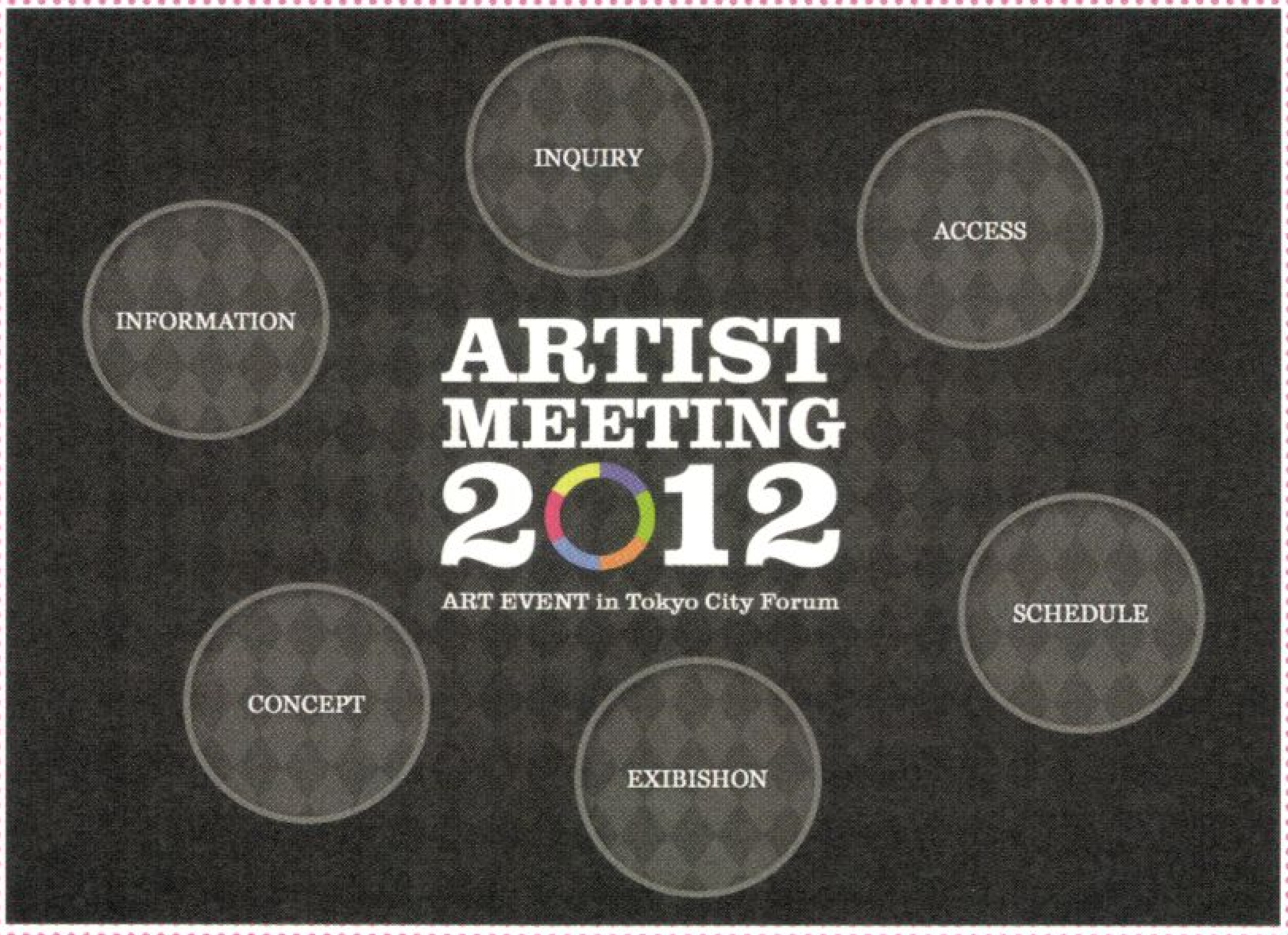
06-3 マウスオーバー時



まとめ [CONCLUSION]

コンテンツ表示エリアが縦に長いWebサイトの場合、ヘッダーにあるメニューが固定されていると、ページの移動がしやすく使いやすいページになります。
そこにちょっとした動きを加えることも、CSS3を使うだけで可能になります。今回使用したtransformとtransitionのプロパティを併用することで、今までになかった、一風変わった動きを実現することができるようになりました。

04 ウィンドウサイズに連動するリキッドタイプのフリーレイアウト



ウィンドウサイズに連動して配置が調節されるリキッドタイプの特性を残しながら、絶対配置を使ったフリーなレイアウトを実現します。ポイントは座標の指定に%を用いること。できるだけ幅広い画面サイズに対応させたい場合に有効なテクニックです。

制作・文／高橋としゆき（Graphic Arts Unit）

制作のポイント

- 絶対配置の座標を%で指定する
- 背景画像にグラデーションを重ねて奥行きを出す
- border-radiusでボタンを円形にする

対応ブラウザ

IE	7 以上	Safari	5 以上
Firefox	3 以上	Chrome	12 以上

※IEは一部の処理に非対応。

TECHNIC 1

ランダムに配置したボタンをリサイズに連動させる

ナビゲーションボタンをウィンドウ幅の変化に連動させることで、レイアウトを多くの画面サイズに対応させます。





TECHNIC 2

円形グラデーションで周辺の明度を落とす

中央部分が明るくなるようにグラデーションを重ねることで、メインのタイトルを引き立たせます。





パーツを準備する

01

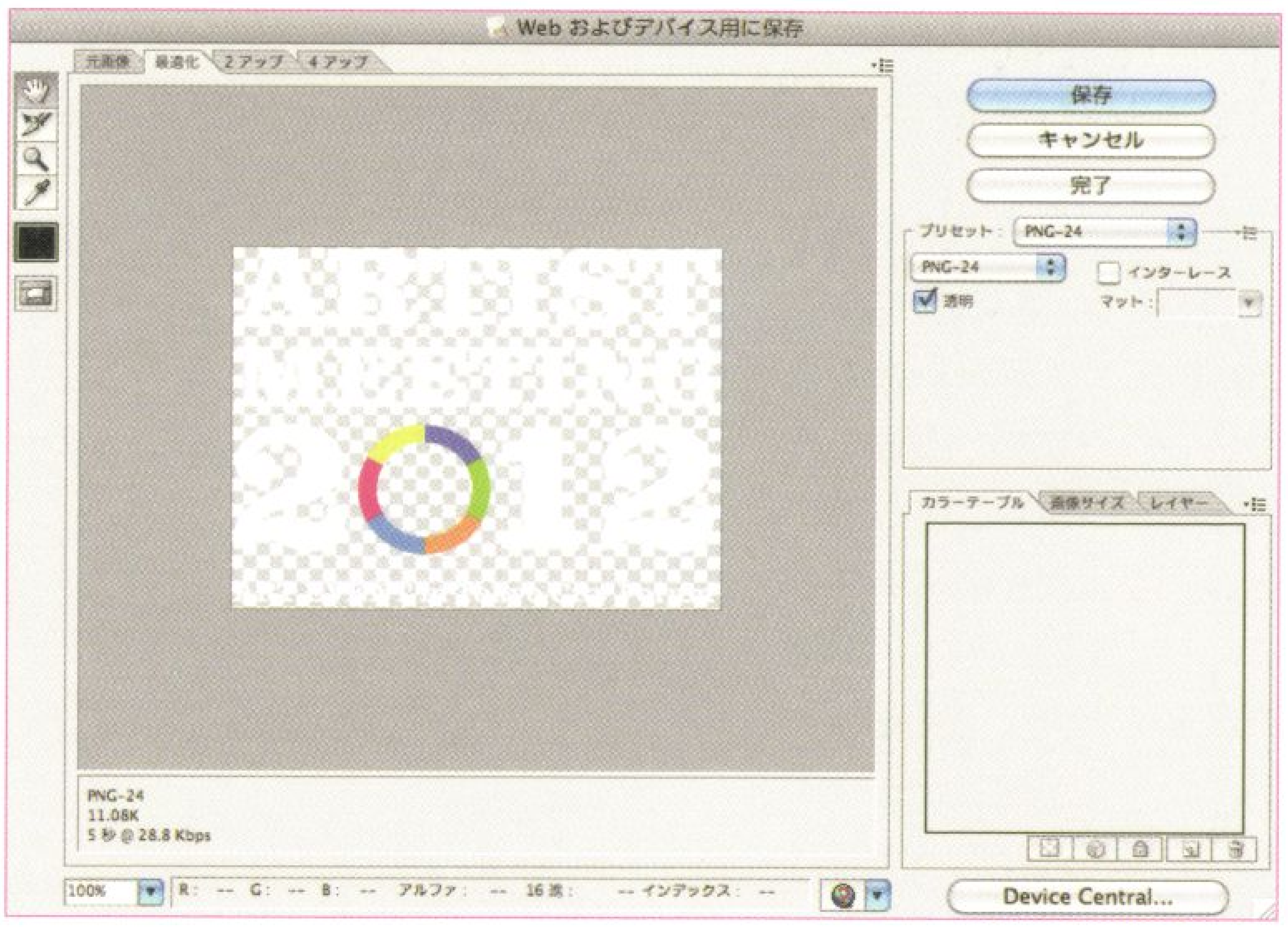
まずはIllustratorを使って01-1のようなタイトル画像を作ります。このパーツは「ファイル」メニュー→「Webおよびデバイス用に保存...」を選択し、[プリセット：PNG-24]で透過pngとして書き出しておきましょう01-2。

> **MEMO**
>
> 01と02ではMac OS X版のIllustrator CS5とPhotoshop CS5を使用していますが、画像を同じような状態で保存できれば、ほかのデザインツールや画像処理ソフトを使っても問題ありません。

01-1 タイトル画像



01-2 [プリセット：PNG-24] で書き出す



02

続いてPhotoshopで02-1のような背景のパターンを作成してJPEG形式で保存します。さらに、これを02-2のように少し明るく補正し、同じくJPEG形式で保存します。先に保存した方がページ全体の背景画像、後の方がナビゲーションボタン内の背景画像です。

02-1 背景のパターン(ページ全体)



02-2 背景のパターン(ナビゲーションボタン内)



ベースとなるHTML／CSSの記述

03

続いてHTMLのコーディングです。03-1のように、タイトルh1要素、ナビゲーションをul要素でマークアップします。タイトルとナビゲーションは、「content」というidのdiv要素で囲んでおきます。また、ul要素には「nav」というidを付加しました。

03-1 ベースとなるHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
― 省略 ―
<body>
<div id="content">
  <h1><img src="images/logo.png" width="300" height="230" alt="ARTIST MEETING 2012" /></h1>
  <ul id="nav">
    <li><a href="info.html">INFORMATION</a></li>
    ― 省略 ―
    <li><a href="inquiry.html">INQUIRY</a></li>
  </ul>
</div>
</body>
</html>
```

04

続いてCSSのコーディングに移りますが、その前にIE6、7、8のみCSSセレクタを拡張するスクリプトを読み込むため、HTMLのhead要素に04-1のコードを追記します。

「selectivizr.js」04-2はIE8以前でサポートされていないCSSセレクタを使えるようにするものです。jQueryやMooTools 04-3などのJavaScriptライブラリと合わせて使います。今回はMooToolsを利用しました。

04-1 HTMLのheadでJavaScriptライブラリを読み込む

```
<!--[if (gte IE 6)&(lte IE 8)]>
  <script type="text/javascript" src="js/mootools-core-1.4.2-full-nocompat.js"></script>
  <script type="text/javascript" src="js/selectivizr.js"></script>
<![endif]-->
```

04-2 selectivizr.js

http://selectivizr.com/

04-3 MooTools

http://mootools.net/

MEMO

「selectivizr.js」は、組み合わせて使うJavaScriptライブラリによって、サポートするセレクタが異なります。今回使うMooToolsとの組み合わせは、jQueryと組み合わせるよりも多くのセレクタを利用できます。

05

続いて「style.css」という空のテキストファイルを作成し、HTMLのhead要素に05-1の1行を追記して読み込みます。このファイルは、必ず先ほど追記したIE用の記述よりあとに記述しましょう。

05-1 HTMLのheadでCSSファイルを読み込む

```
<link href="css/style.css" rel="stylesheet" type="text/css" />
```



06

「style.css」に06-1を入力します。これは全体に関わるスタイルです。
まず「/*各要素の初期設定*/」以下は、各要素のスタイルを初期化するための処理です。そして「/*全体の背景パターン*/」以下で、ページ全体の背景画像を指定します。
背景画像の配置が「fixed」になっているのは、ナビゲーションボタンの背景画像と位置を合わせるための処理です。

06-1 全体に関わるスタイル（style.css）

```
@charset "UTF-8";

/*各要素の初期設定*/
body, h1, div, ul, li, a {
  margin: 0;
  padding: 0;
}
ul, li {
  list-style: none;
}
a {
  text-decoration: none;
  outline: 0;
}

/*全体の背景パターン*/
body {
  background: url(../images/background01.
  jpg) repeat center center fixed;
}
```

ATTENTION!

IEはfixedの解釈が異なるため、bodyの背景画像とナビゲーションボタンの背景画像の位置を合わせることができません。

コンテンツを内包する要素のスタイルを作成する

07

続いて07-1のスタイルを追記します。これは、コンテンツの全体を内包する「#content」のスタイルです。width、heightともに100%なので、ページ全体を覆うように配置されます。さらに、背景は中央部分が透明になるグラデーションを指定します。こうすることで、ページの周辺が暗くなり、中央にスポットライトが当たっているような演出ができます07-2 07-3。
また、ブラウザのサイズが小さくなったとき要素同士が重ならないよう、6、7行目のmin-widthとmin-heightで最大幅、最小幅を決めています。

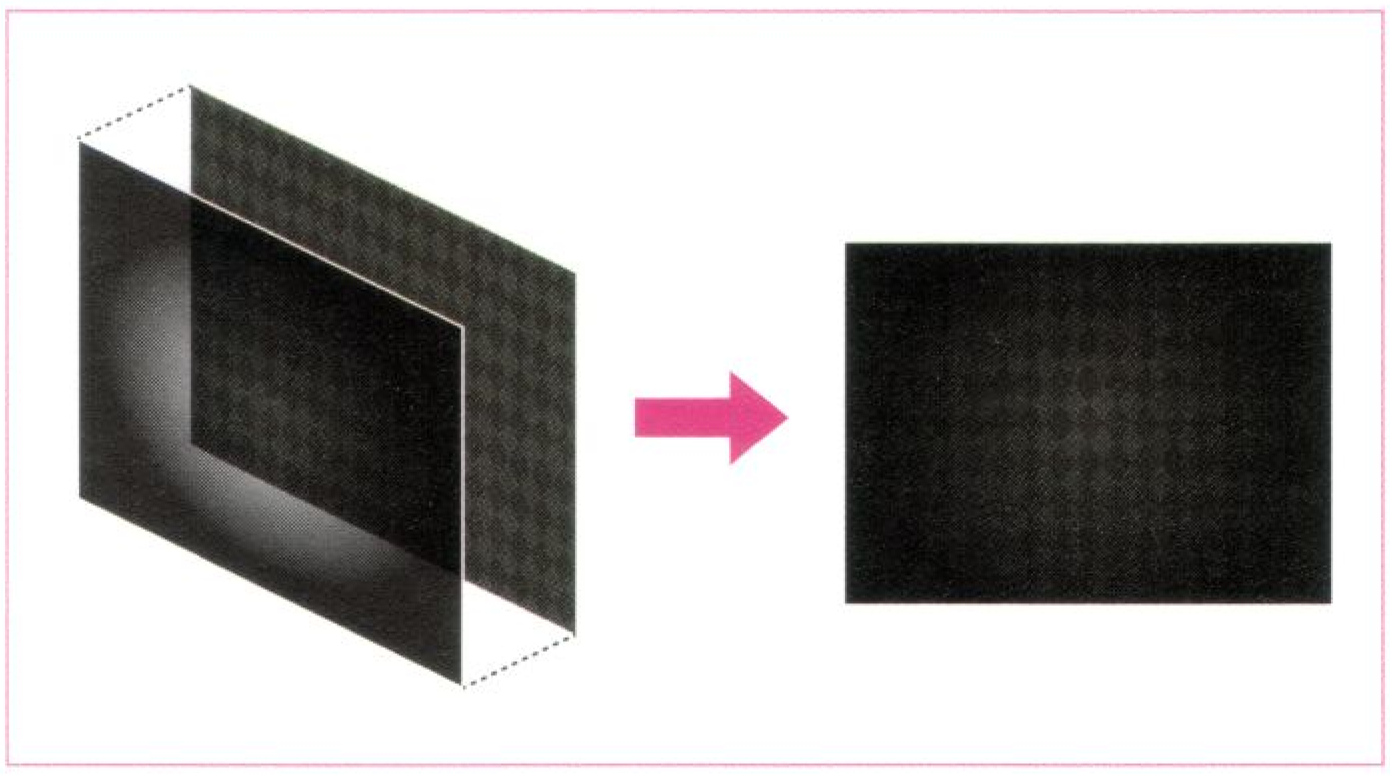
07-1 コンテンツを内包する要素のスタイル（style.css）

```
/*コンテンツを内包する要素*/
#content {
  position: absolute;
  width: 100%;
  height: 100%;
  min-width: 800px;
  min-height: 680px;
  background: -webkit-radial-gradient(center, rgba(0, 0, 0, 0) 25%, rgba(0, 0, 0, 1) 100%);
  ― 省略 ―
  background: radial-gradient(center, rgba(0, 0, 0, 0) 25%, rgba(0, 0, 0, 1) 100%);
}
```

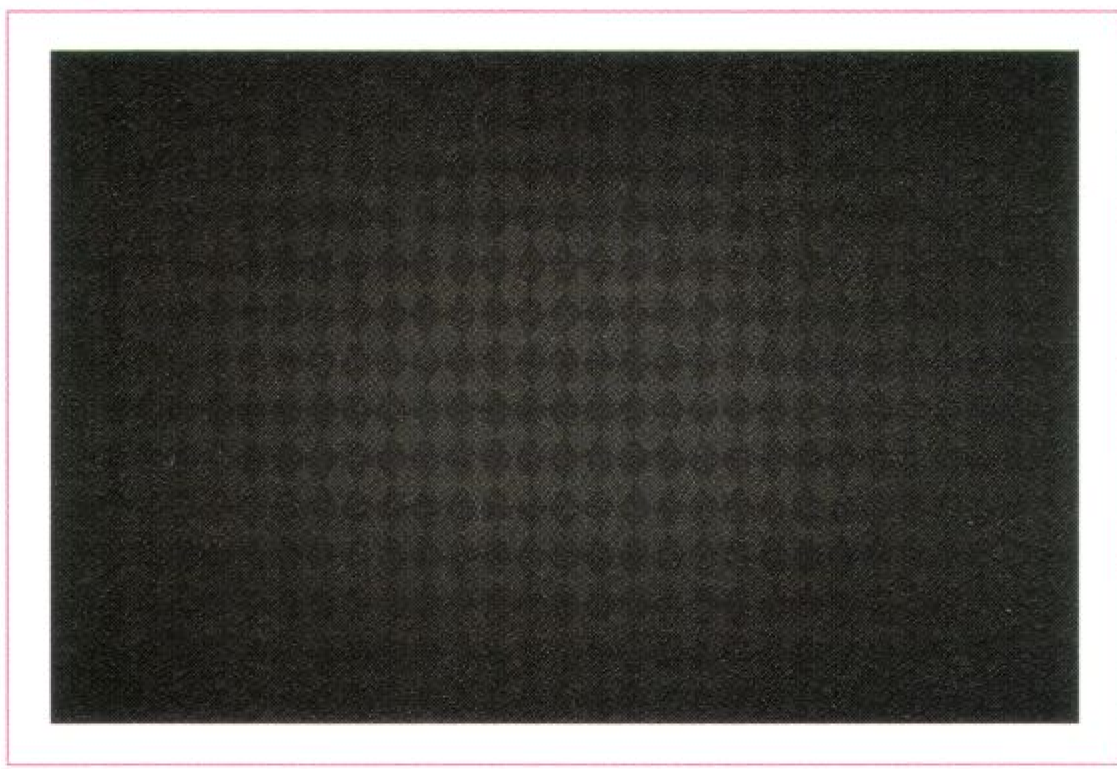
ATTENTION!

IEはグラデーションに対応していないので、通常のパターンのみとなります。

07-2 背景は中央部分が透明になるグラデーションとする



07-3 中央にスポットライトが当たっているような演出



タイトルのスタイルを作成する

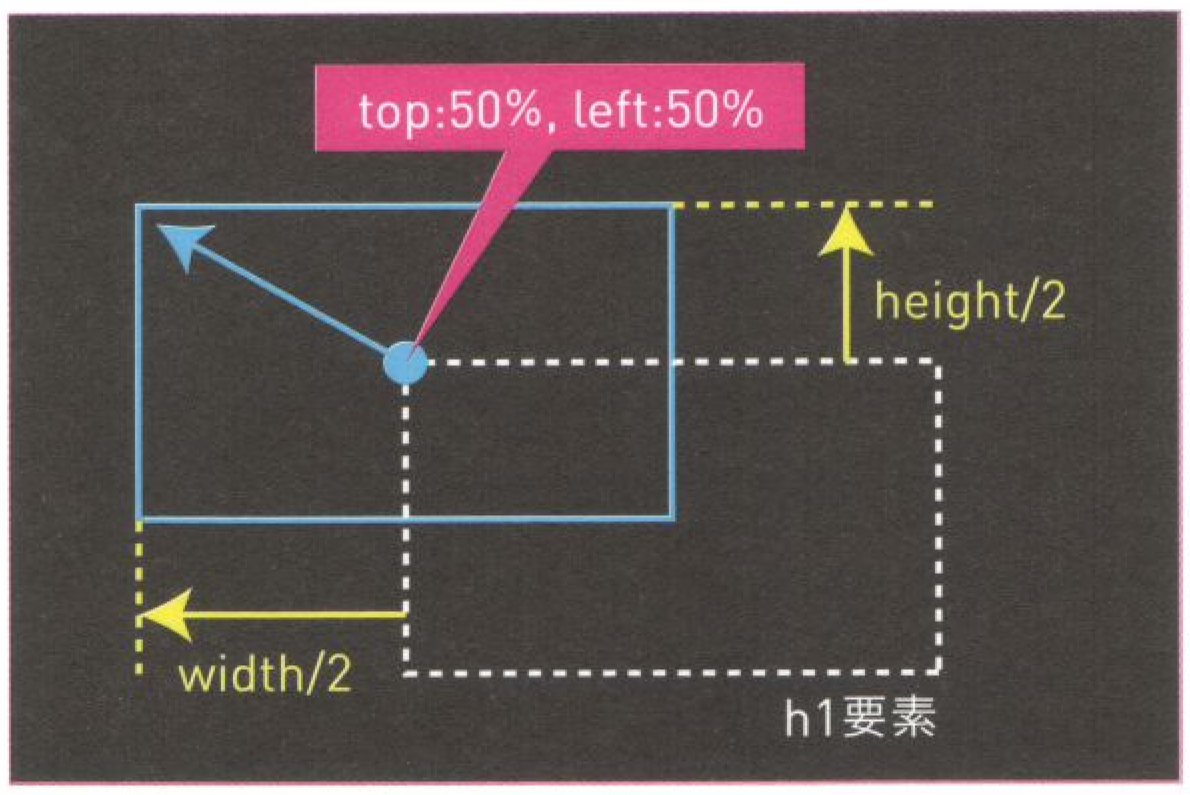
08

さらに08-1のスタイルを追記します。leftとtopを50％に設定し、ロゴのサイズの半分をネガティブマージンとして指定することで、ロゴが常に「#content」の中央に表示されるようになります08-2 08-3。

08-1 ロゴ（タイトル）に関するスタイル（style.css）

```
/*タイトル*/
#content h1 {
  position: absolute;
  top: 50%;
  left: 50%;
  width: 300px;
  height: 230px;
  margin: -115px 0 0 -150px;
}
```

08-2 ロゴの配置



08-3 ロゴが中央に表示される



ナビゲーションボタンのスタイルを作成する

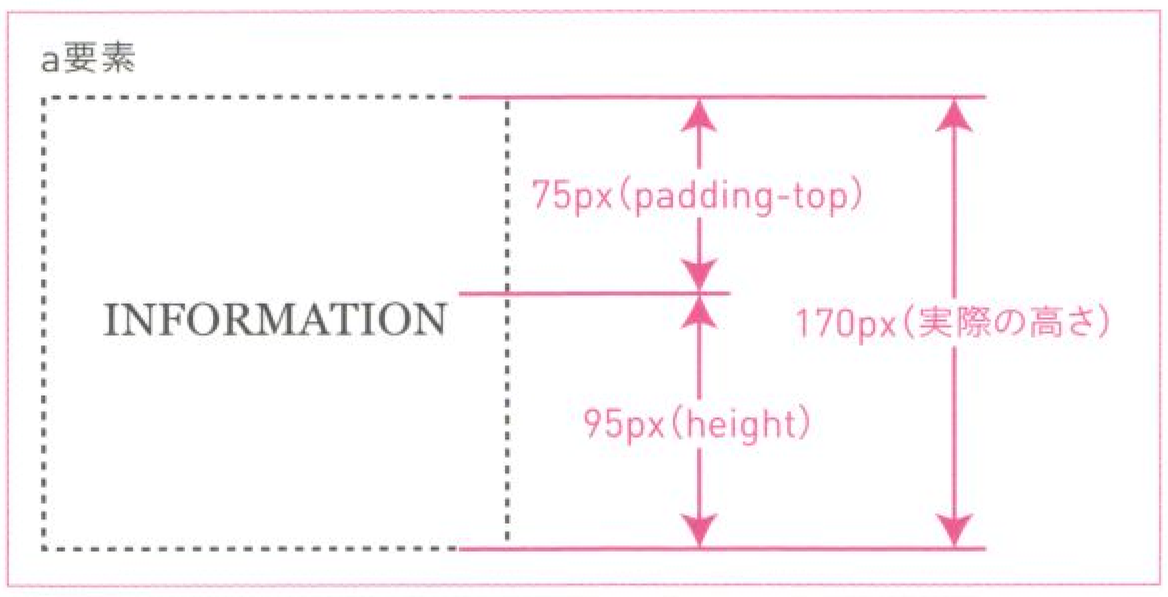
09

次はナビゲーションボタンの共通スタイルとなる09-1を追記します。ナビゲーションボタンはli要素内のa要素をブロック要素として扱い、absoluteの絶対配置でレイアウトしていきます。ここではa要素の扱いやサイズ、フォント、背景画像などを指定します。09-2のように、ボックスの実際の高さが上部paddingとheightの合計になることに注意しましょう。また、背景画像（background）はbodyの背景と同じように「fixed」を指定し、位置が常に背景と揃うようにします。

09-1 ナビゲーションボタンの共通スタイル（前半／style.css）

```
/*ナビゲーションボタン共通スタイル*/
#nav li a {
  position: absolute;
  display: block;
  width: 170px;
  height: 95px;
  padding-top: 75px;
  background: url(../images/background02.jpg) repeat center center fixed;
  font-family: Georgia, "Times New Roman", Times, serif;
  font-size: 18px;
  color: #fff;
  text-align: center;
  ― 省略（10で解説） ―
}
```

09-2 ボックスの実際の高さが上部paddingとheightの合計になる





10

続いて10-1では、ボックス自体のシャドウ、テキストのシャドウを設定します。box-shadowではinsetを指定して内側にシャドウを施しています10-2。11～13行目は、ボックスを円形にするための処理です。円形のボックスを作るには、このようにborder-radiusに50％を指定します。角丸の半径がボックスの幅、高さの半分の値になり、自動的にボックスを円形にすることが可能です10-3。14行目は、IEでborder-radiusを有効にするため「PIE.htc」10-4を読み込んでいます。これで基本のスタイルは設定完了です。

10-1 ナビゲーションボタンの共通スタイル（後半／style.css）

```
01 /*ナビゲーションボタン共通スタイル*/
02 #nav li a {
03   ― 省略（09で解説） ―
04   -webkit-box-shadow: inset 3px 3px 10px rgba(0, 0, 0, 0.5);
05   ― 省略 ―
06   box-shadow: inset 3px 3px 10px rgba(0, 0, 0 ,0.5);
07   -webkit-text-shadow: 0 0 5px rgba(0, 0, 0, 1);
08   ― 省略 ―
09   text-shadow: 0 0 5px rgba(0, 0, 0, 1);
10   border: 5px solid #6c7376;
11   -webkit-border-radius: 50%;
12   ― 省略 ―
13   border-radius: 50%;
14   behavior: url(../css/PIE.htc); /*動作させるには絶対パスに修正する*/
15 }
```

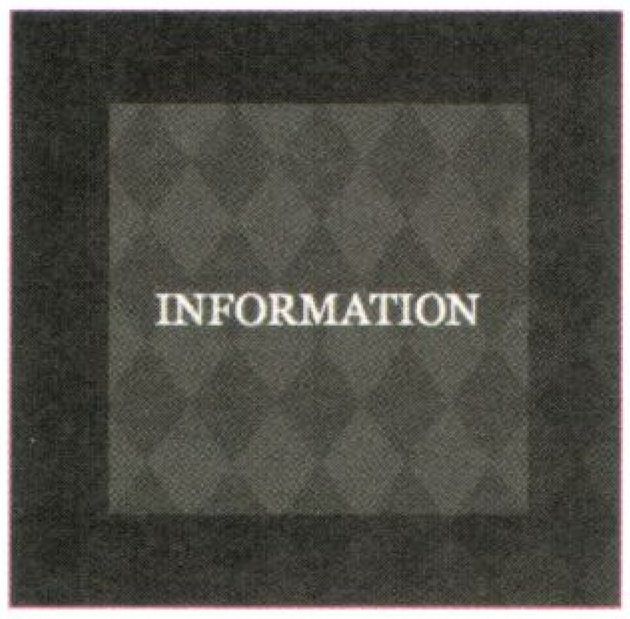
MEMO

PIE.htcは、IEがサポートしていないCSS3プロパティの一部をIEにも対応させるためのビヘイビアです。このファイルをCSSから読み込むことで「border-radius」、「box-shadow」、「linear-gradient」のプロパティをIEでも利用できます。使用の際は、ベンダープレフィックスとして「-pie-」を付加した上で、通常のプロパティと同じ要領で指定します。

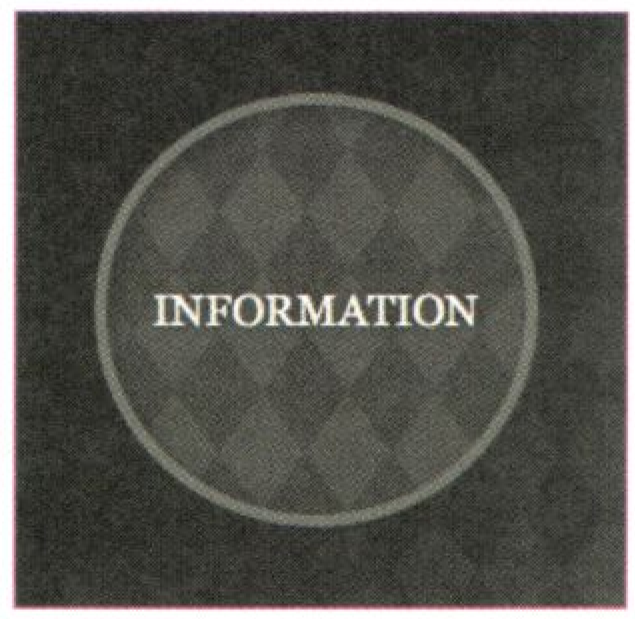
ATTENTION!

紙面ではPIE.htcを相対パスで指定していますが、実際にはhttpから始まる絶対パスで指定する必要があります。なお、PIE.htcはinsetのシャドウに対応していないため、IEではシャドウが表示されません。

10-2 ボックス型のシャドウ



10-3 ボックスを円形にした



10-4 PIE.htc

http://css3pie.com/

11

続いて11-1を追記します。これらは、ナビゲーションボタンごとに適用する個別スタイルです。HTML側のli要素に個別のclass名を付けてもいいのですが、今回はHTMLをできるだけシンプルにするため:nth-child擬似クラスを使って要素を選択します。:nth-childに続く()内に任意の数字を入れることで、その順番の要素を選択できます11-2。

今回は6個のli要素があるので、1から6までそれぞれの座標を％で指定しています。

IE8以前は本来この擬似クラスをサポートしていませんが、「selectivizr」を使って対応させています。

ブラウザで表示させると、11-3のようになります。

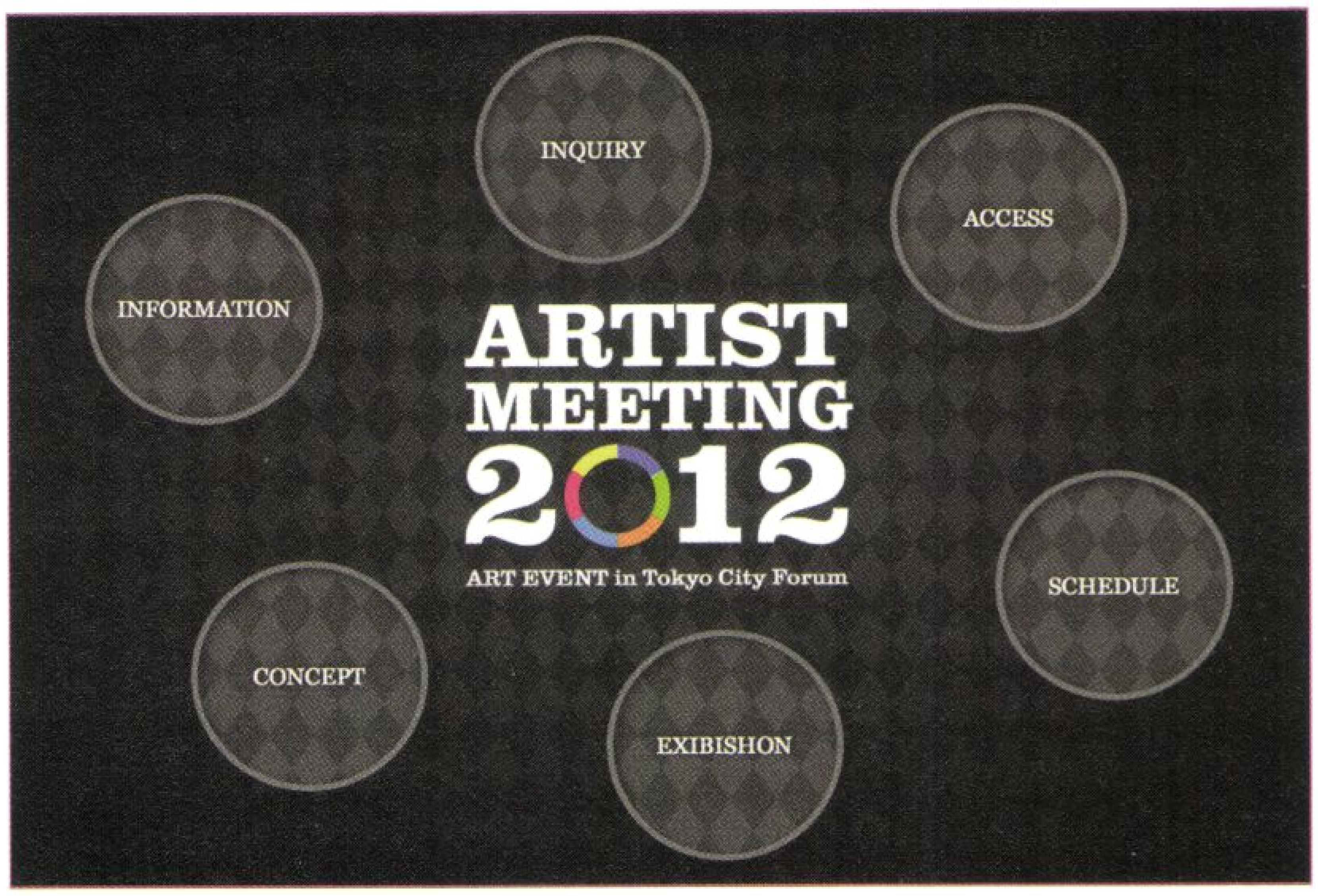
11-1 ナビゲーションボタンの個別スタイル（style.css）

```
/*ナビゲーションボタン個別スタイル*/
#nav li:nth-child(1) a {
  top: 21%;
  left: 6%;
}
#nav li:nth-child(2) a {
  bottom: 11%;
  left: 14%;
}
#nav li:nth-child(3) a {
  bottom: 3%;
  right: 36%;
}
#nav li:nth-child(4) a {
  bottom: 21%;
  right: 6%;
}
#nav li:nth-child(5) a {
  top: 11%;
  right: 14%;
}
#nav li:nth-child(6) a {
  top: 3%;
  left: 36%;
}
```

11-2 :nth-child擬似クラスによる要素選択

```
<ul>
    <li>list-1</li> ・・・・・・・・・・ :nth-child(1)
    <li>list-2</li> ・・・・・・・・・・ :nth-child(2)
    <li>list-3</li> ・・・・・・・・・・ :nth-child(3)
    <li>list-4</li> ・・・・・・・・・・ :nth-child(4)
    <li>list-5</li> ・・・・・・・・・・ :nth-child(5)
</ul>
```

11-3 ブラウザで表示させたところ



MEMO

selectivizr.jsは、IEがサポートしていない擬似クラスや擬似要素、属性セレクタなどをIEでも使用可能にするJavaScriptプラグインです。基本的に「jQuery」や「Prototype」「MooTools」などのJavaScriptライブラリと組み合わせて使いますが、組み合わせるライブラリによってサポートされるクラス、要素、セレクタが異なるので注意が必要です。今回のように「MooTools」と組み合わせた場合はすべてを使うことができます。

ATTENTION!

「selectivizr」はローカルでは動作しないことがあるので、サーバーにアップロードしてからの確認が必要です。また、外部CSSしか認識しないため、埋め込みやインラインでの記述では動作しません。

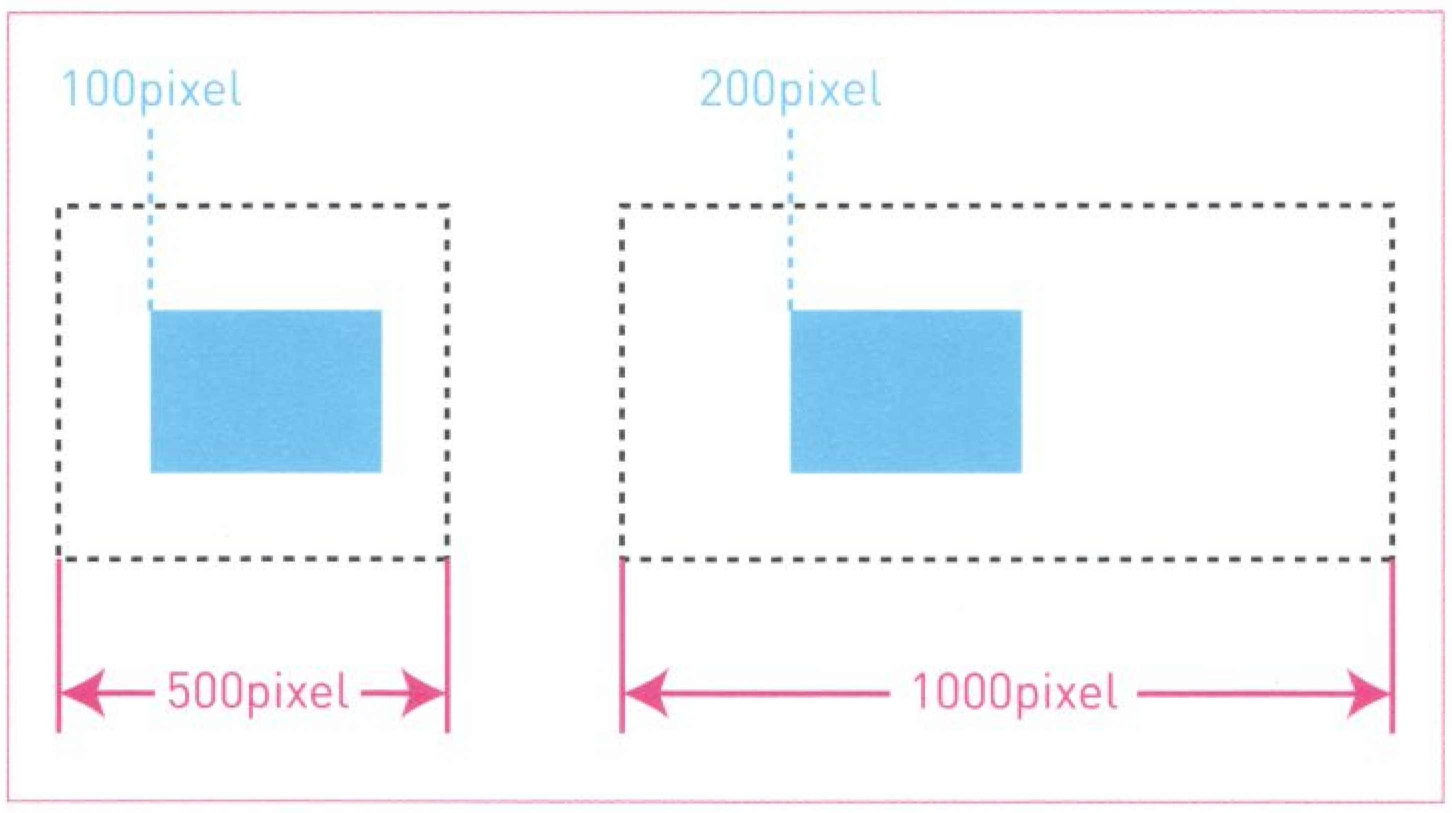
TIPS

:nth-child(odd)と記述すると奇数番目の要素のみ、:nth-child(even)と記述すると偶数番目の要素のみを選択することもできます。これをtableのtr要素に指定すれば、1行おきに色を変えるなどの処理も可能です（222ページのCHAPTER04-03「擬似クラスを利用した可読性の高いテーブル」も参照）。また、数字に続いて「n」を付加すれば、指定した数字ごとの要素を選択できるなど、さまざまな使い方ができます。

12

絶対配置の座標を%で指定すると、親要素の幅、高さに対しての相対値として配置できます。たとえば、leftに20%を指定した場合、親要素の幅が500ピクセルのときは左から100ピクセル、親要素の幅が1000ピクセルのときは左から200ピクセルに配置されるということです 12-1。このサンプルではこれを利用して、ウィンドウサイズに応じたリキッドレイアウトを実現しています。

12-1 leftに20%を指定したときの配置



MEMO

水平方向の座標は、親要素の左を基準としたいときはleft、右を基準としたいときはrightで指定します。垂直方向の座標は、親要素の上を基準としたい場合はtop、下を基準としたい場合はbottomで指定します。



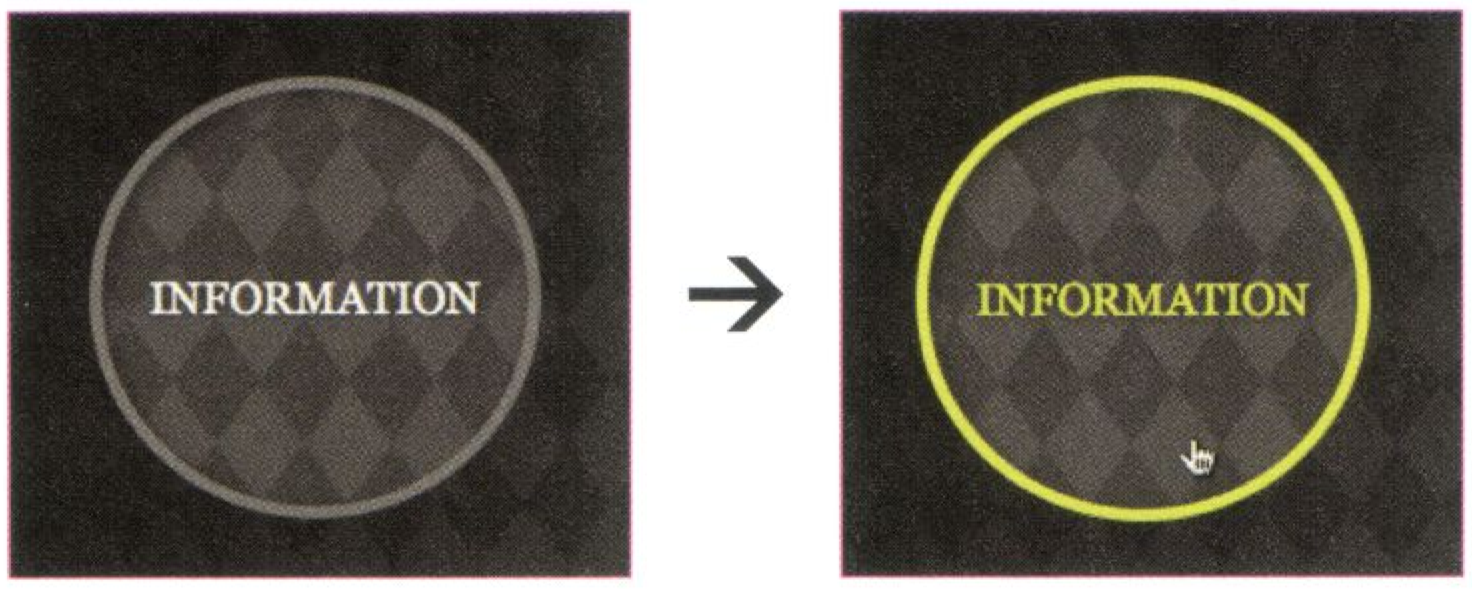
13

最後に 13-1 を追記します。これはボタンにロールオーバーした際のスタイルです。各要素とも、罫線とテキストの色が変化するように設定しました 13-2。

13-1 ロールオーバー時のスタイル（style.css）

```
/*ナビゲーションボタンロールオーバー*/
#nav li:nth-child(1) a:hover {
  color: #ff0;
  border-color: #ff0;
}
#nav li:nth-child(2) a:hover {
  color: #f06;
  border-color: #f06;
}
#nav li:nth-child(3) a:hover {
  color: #39f;
  border-color: #39f;
}
#nav li:nth-child(4) a:hover {
  color: #66c;
  border-color: #66c;
}
#nav li:nth-child(5) a:hover {
  color: #9c0;
  border-color: #9c0;
}
#nav li:nth-child(6) a:hover {
  color: #f90;
  border-color: #f90;
}
```

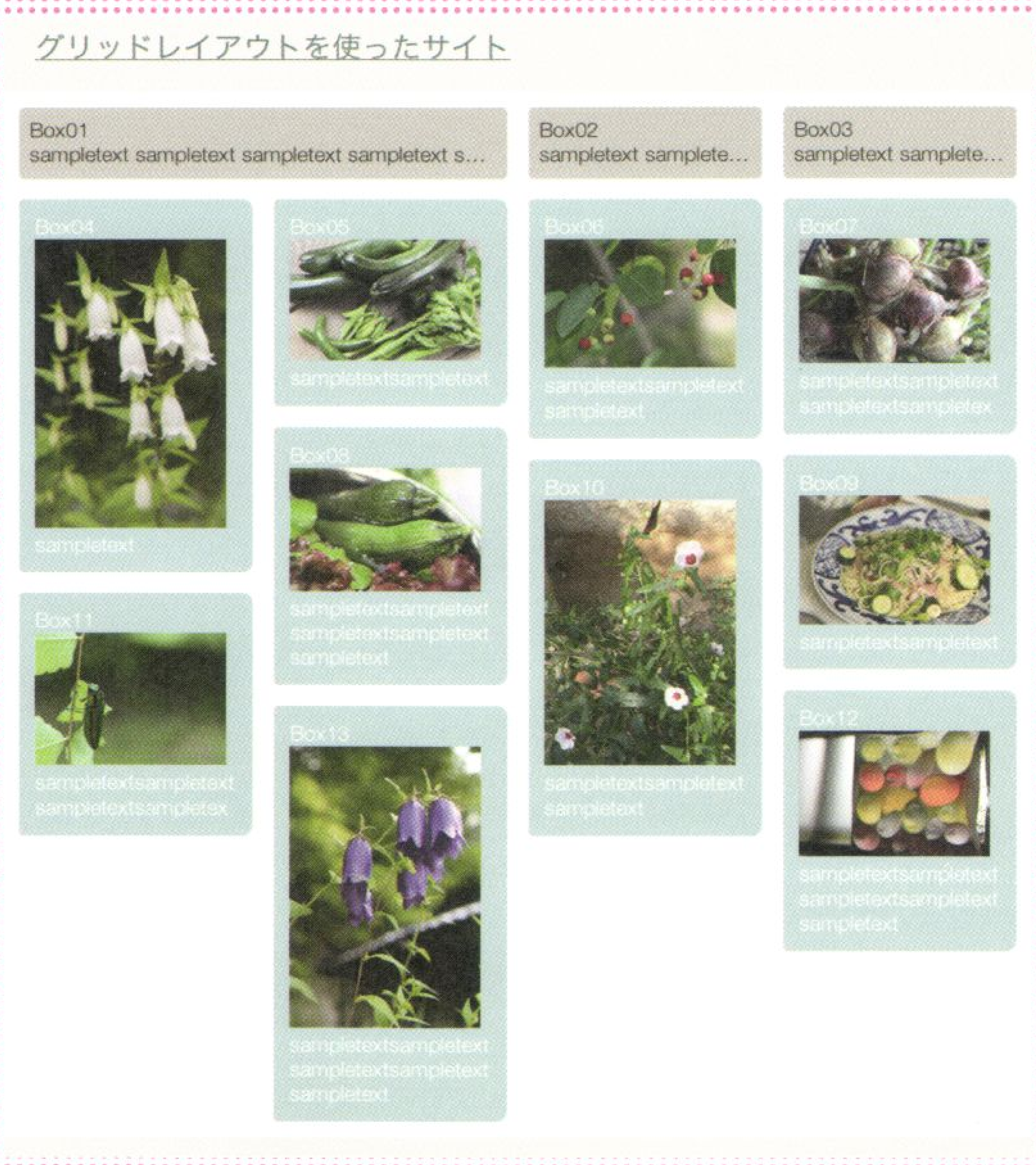
13-2 ロールオーバー時に罫線とテキストの色が変化する



まとめ [CONCLUSION]

absoluteでの絶対配置を使えば要素を正確な位置へレイアウトできますが、想定するウィンドウサイズ以外では要素が隠れたりスペースが空きすぎることがあります。しかし、今回のように座標指定に%を使えば、リサイズに追従するリキッドレイアウトの長所を取り入れることができます。目的や用途を考えながら、可能であれば積極的に活用しましょう。

05 グリッドレイアウトで実現するレスポンシブなサイト



プラグインを利用して、レスポンシブWebデザインにも対応した、グリッドレイアウトを簡単に作る方法を説明します。CSS3を使うことで、より簡単に複数のボックスを使ったデザインを作ることができます。

制作・文／藤本勝己（株式会社レリーフ）

制作のポイント

- グリッドレイアウトを簡単に実装
- 複雑になりがちな角丸をborder-radiusでシンプルなコードにする
- 長いURLの表示などを、word-wrapで問題回避

対応ブラウザ

IE	9 以上	Safari	5.1 以上
Firefox	9 以上	Chrome	16 以上

※IE7 ～ 8はborder-radiusに非対応。

TECHNIC 1

ボックスの横幅を揃える

プラグインを利用することで、各ボックスの高さを揃えなくても自動的に積み重なったレイアウトを作ることができます。横幅はボックスごとの幅を揃えていかないと、隙間ができてしまう点に注意しましょう。



TECHNIC 2

長いテキストは省略して表示

text-overflowプロパティを利用して、テキストが表示領域の横幅をオバーフローした際には自動的に略し、省略記号を表示するよう設定しています。





グリッドレイアウトの準備

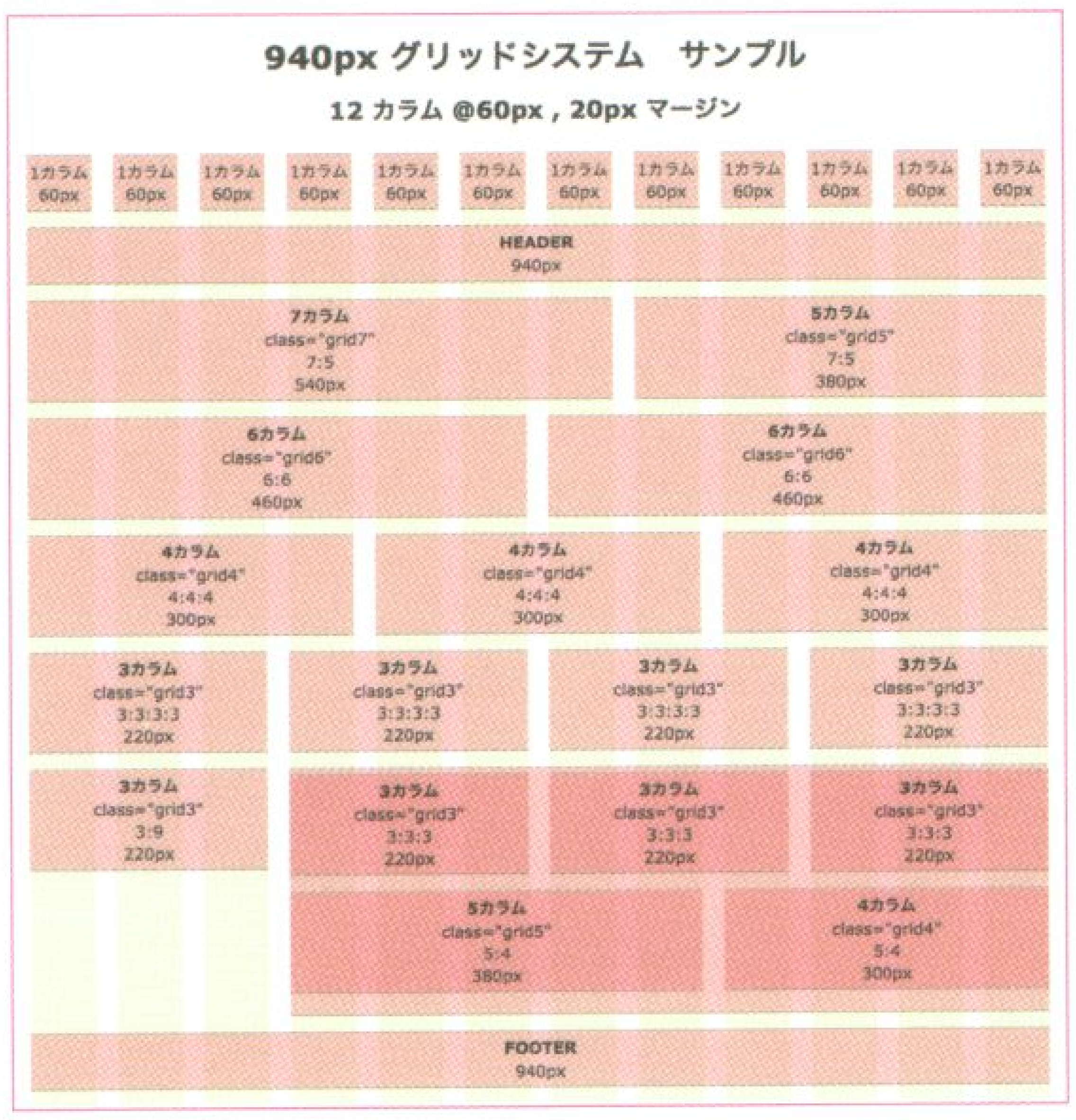
01

まず、横幅、縦幅をコンテンツの内容ごとに決めていきます。3列になった場合、4列になった場合はどうなるだろう、と想定しながら決める必要があります。
グリッドレイアウトの設計を決める際は、URL 01-1 からアクセスした図 01-2 を参考にしながら考えるとわかりやすいです。

01-1 「Webデザインレシピ」940pxグリッドシステムのサンプル

http://webdesignrecipes.com/940grid.html

01-2 グリッドシステムの参考図



02

次に、「jQuery」 02-1 とjQueryプラグイン「jQuery Masonry」を入手します。jQuery Masonryは、雑誌のような段組レイアウトを簡単に作成することができるJavaScriptです。
Masonryは、02-2 のURLにアクセスし、「Download the script jquery.masonry.min.js」と書かれたリンクをクリックして、ダウンロードを行います 02-3。ダウンロードが済んだら、jQuery本体と同じフォルダなどに置いておきましょう。

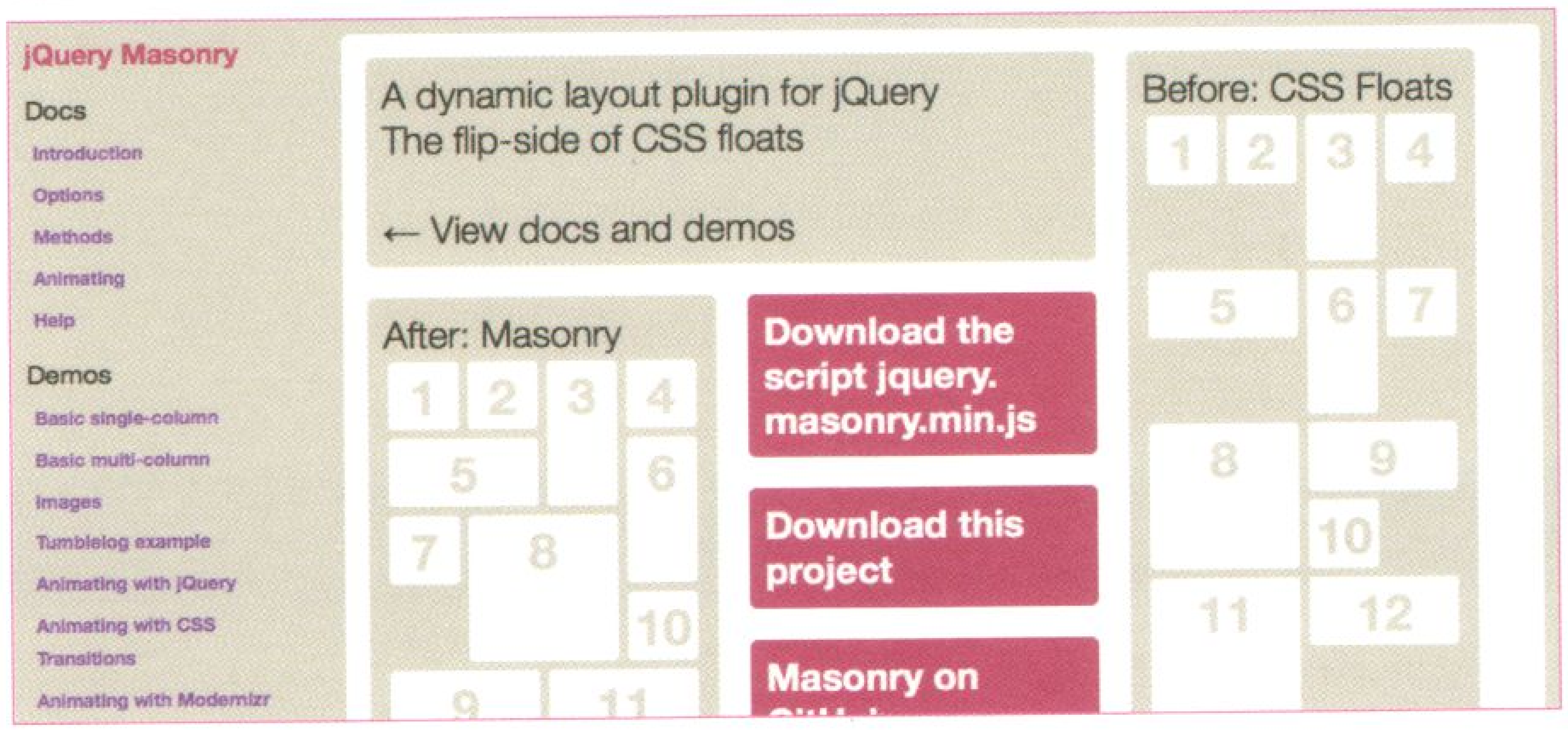
02-1 jQuery

http://jquery.com/

02-2 Masonryプラグインの公式サイト

http://masonry.desandro.com/

02-3 「Download the script jquery.masonry.min.js」をクリックしてダウンロード



03

<body>～</body>タグの間に、03-1のようにスクリプトを記述します。このスクリプトは「jquery.js」と「jquery.masonry.min.js」を読み込み、masonryメソッドを呼び出すものです。
そのほか利用できるプロパティなどは、02で紹介したjQuery Masonryのサイトで詳しく説明されています。

03-1 Masonryメソッドを呼び出すスクリプト

```
<script src="js/jquery-1.7.1.min.js"></script>
<script src="js/jquery.masonry.min.js"></script>
<script>
  $(function(){
    var $container = $('#container');
    $container.masonry({
      itemSelector: '.item',
      columnWidth: 240,
      isAnimated: !Modernizr.csstransitions
    });
    // dynamically load sites using Masonry from Zootool
    $.getJSON('http://zootool.com/api/users/
items/?username=desandro' +
      '&apikey=8b604e5d4841c2cd976241dd90d319d7' +
      '&tag=bestofmasonry&callback=?')
     .error( ajaxError )
     .success(function( data ){
     });
  });
</script>
```

04

水色のボックスで設定したCSSは04-1のようになります。
classをそれぞれ個別に設定することで、横幅を変えることもできます。

04-1 水色のボックスで設定したCSS

```
.homepage .item {
  width: 200px;
  float: left;
  padding: 10px;
  margin: 10px;
  font-weight: 300;
  ― 省略 ―
  background-color: #CCCCCC;
}
```

MEMO

ボックスのwidthを個別に変更する場合、a属性に共通でwidthを指定していますので、注意してください01。

01 a属性に共通でwidthを指定

```
.homepage .link a {
  display: block;
  padding: 15px;
  width: 190px;
  background: #BEDAE4;
  color: #FFFFFF;
  /* 角丸 */
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 10px;
}
```

border-radiusでボックスの角を丸くする

05

各ボックスをborder-radiusを使って角丸に設定します。border-radiusプロパティは、4つのborder-＊-radiusを一括で指定することができます05-1。
このサンプルでは、05-2のようにborder-radiusに値を1つだけ指定しています。この場合は、4つの角に同じ値が適用されます。



05-1 border-radiusの基本的な記述

```
border-radius : 5px 10px 20% 30px;
```

- 5px：左上角丸の半径
- 10px：右上角丸の半径
- 20%：右下角丸の半径
- 30px：左下角丸の半径

05-2 border-radiusの記述

```
.homepage .item {
  — 省略 —
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
  — 省略 —
}
```

TIPS

border-radiusの8つの値を変えることで、四方の角丸のサイズを個別に設定できます01。

01 四方のサイズをそれぞれ指定する

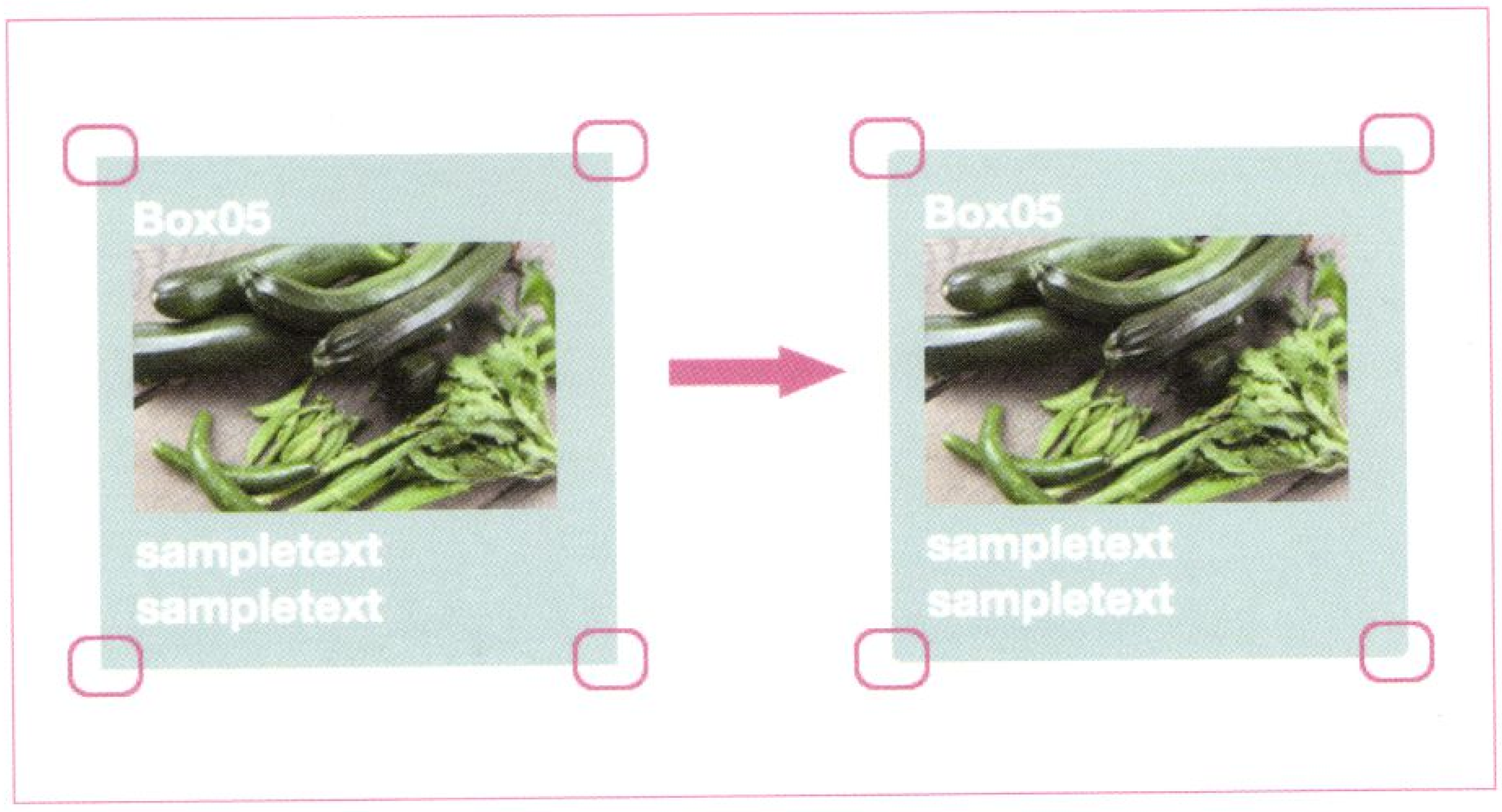



```
               ①    ②    ③    ④
border-radius: 10px 20px 30px 40px /
               15px 25px 35px 45px
               ⑤    ⑥    ⑦    ⑧
```

06

IE7～8はborder-radiusに対応していないため、IE7やIE8で表示した場合には角が四角くなりますが、アクセシビリティは確保できます。
iOSのSafariでは、説頭辞として-webkit-が必要ですので、表示を求められる場合には、併記する必要があります。
Firefox 2.0では、背景画像を表示させているボックスの角は丸くなりません 06-1。

06-1 Firefox 2.0での表示



word-wrapで単語の折り返しを設定

07

word-wrapプロパティは、文字数が多い英単語やURLが1行に収まり切らない場合、文字列の途中で改行するかどうかを指定するプロパティです。
値にはnormal、break-wordの2つのキーワードのうち、いずれかを指定することができ、これによりURLや文字列が多い単語の扱いを決めることができます 07-1。

07-1 word-wrapの基本的な記述

```
word-wrap: break-word;
```

- break-word：キーワードの指定

08

CSSの記述は、08-1のように書きます。word-wrap:break-wordを指定すると、08-2のように英文の途中でスペースが入った場合でも、文章を折り返すことができます。

08-1 word-wrapの記述

```
.homepage .link {
    padding: 0;
   /* 長い英単語を強制的に折り返す */
    word-wrap: break-word ;
    -moz-word-wrap: break-word ;
}
```

MEMO

word-wrap:normalと指定した場合は、折り返しなしの設定になります。

08-2 文章が折り返されている



09

word-wrapが何も指定されていなければ、09-1のように文字がはみ出して表示されます。

09-1 文字がボックスからはみ出てしまう



text-overflowを設定

10

text-overflowプロパティを使うと、テキストが表示領域をオーバーフローした際に、省略記号("…")を表示する設定ができます10-1。
オーバーフローを強制的に発生させて省略記号を表示するには、要素にいくつかの追加のプロパティを指定する必要があります。以下の例を参照してください。

10-1 text-overflowの基本的な記述

```
text-overflow: ellipsis;
```

ellipsis：「…」を表示

TIPS

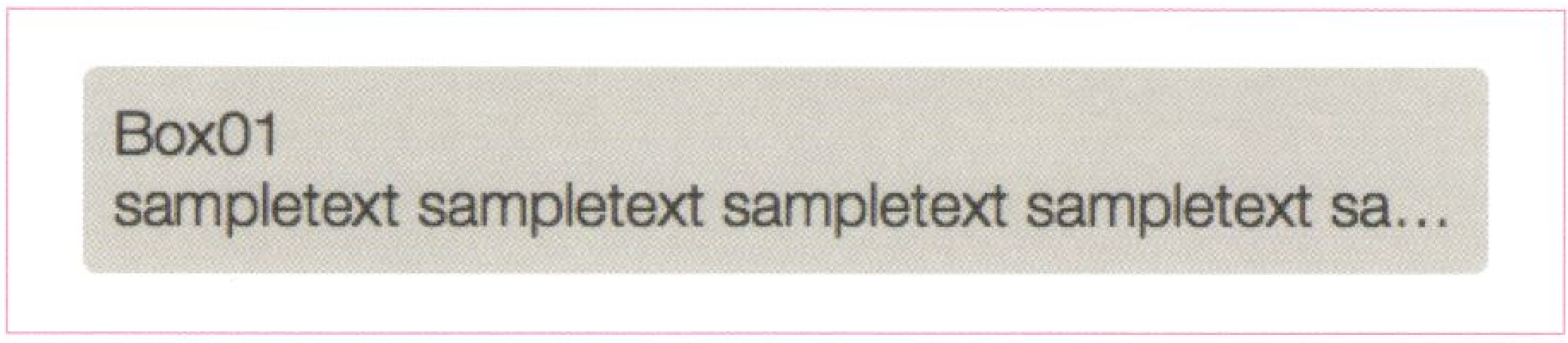
ellipsis	切り取られたテキストの代わりに、省略記号「…」を表示します。
clip	デフォルトの値です。オーバーフローしたテキストを切り取ります。



11

実際に使用したのが 11-1 になります。ボックスの高さが決まっており、長いテキストを表示しきれない場合、text-overflowを使うと便利です。

11-1 オーバーフローしたテキストを省略



12

スタイルシートとHTMLの記述は 12-1、12-2 のようになります。

12-1 スタイルシートの記述

```
.homepage .big-text{
  height:50px;
  /*表示しきれないテキストを…で表示させる */
  text-overflow:ellipsis;
  white-space:nowrap;
  -o-text-overflow:ellipsis;
  overflow:hidden;
}
```

12-2 HTMLの記述

```
<div class="item big-text col2">Box01<br />
sampletext sampletext sampletext sampletext sampletext sampletext sampletext
sampletext sampletext
</div>
```

MEMO

text-overflowだけでは全ブラウザに対応できません。01のように設定しています。

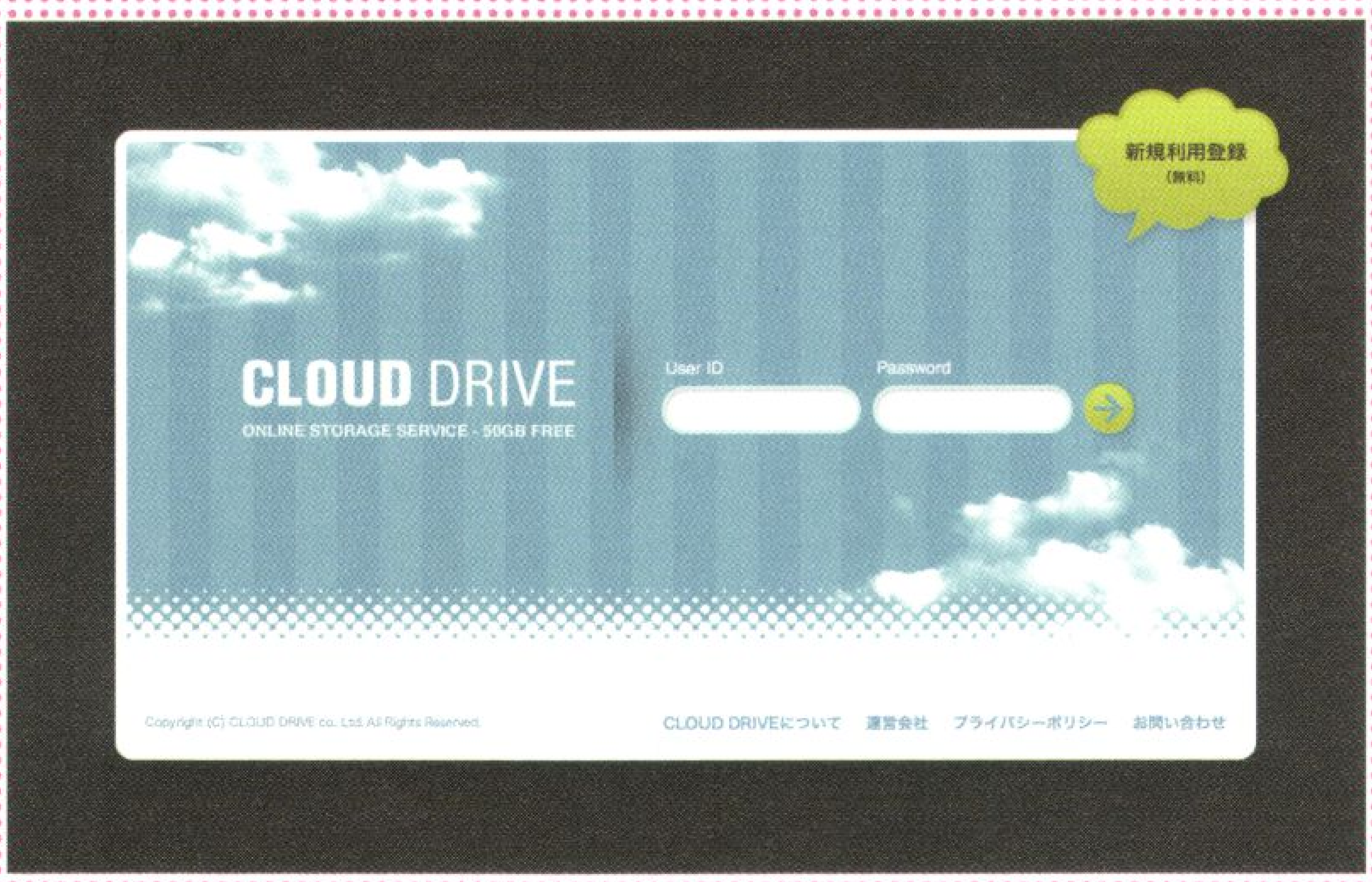
01 text-overflowと合わせて設定

```
text-overflow:   ellipsis;           /* IE, WebKit (Safari, Chrome), Firefox 7, Opera 11 */
white-space: nowrap;
-o-text-overflow: ellipsis;          /* Opera 9-10 */
overflow: hidden;                    /* "overflow" の値は "visible" 以外のものを指定する */
```

まとめ [CONCLUSION]

グリッドレイアウトは難しそうに見えますが、jQueryとCSS3を使用することで、誰でも気軽に実装することができます。
ただし、IEではまだサポートされていないプロパティも多いので、使う際にはIEでの見え方にも注意してください。
慣れてくれば、jQueryを自分でカスタマイズしたり、ボックスの装飾などでオリジナリティを出していきましょう。

06 透過画像を効果的に組み合わせたデザイン



透過画像を効果的に使えば、従来は難しかった複雑なデザインを作ることができます。複数の背景画像や画像を使ったボーダーなど、今後CSS3を使う上でスタンダードになってくるであろう新しい技術を使い、魅力あるレイアウトを効率的に作成しましょう。

制作・文／高橋としゆき（Graphic Arts Unit）

制作のポイント

- border-imageを利用してボーダーに画像を使う
- 背景画像を複数指定組み合わせる
- 負の値を指定してボタンを枠から飛び出して配置

対応ブラウザ

IE	P.E.	Safari	5 以上
Firefox	3 以上	Chrome	12 以上

※IE6は非対応。IE7 ～ 9は一部の処理に非対応。

TECHNIC 1

ボタンを枠から飛び出させる

フォーマットから飛び出したボタンは注視度の高い表現となります。

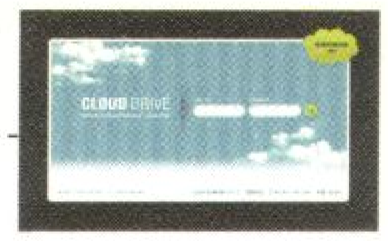


TECHNIC 2

色の境界をハーフトーンパターンで装飾

フッタの上部にハーフトーンパターンの透過画像を用いることで、個性的なグラデーション処理にします。





パーツを準備する

01

まずは雲のパーツを作りましょう。雲のパーツは空の写真から雲の部分だけを抽出して利用します。
空の写真をPhotoshopで開き、必要な雲の部分だけを残して画像を切り抜きます。「イメージ」メニュー→「色調補正」→「彩度を下げる」を実行して 01-1 のようにグレースケールにしたあと、「イメージ」メニュー→“色調補正”→“レベル補正…”で画像のコントラストを高めに補正します 01-2。雲のディティールを損なわない程度で、空の部分が完全な黒になるように調節しましょう。どうしても黒くならない部分があれば、ブラシ等を使って塗りつぶしても構いません 01-3。

MEMO
ここではMac OS X版のPhotoshop CS5を使用していますが、画像を同じような状態で保存できれば、ほかのデザインツールや画像処理ソフトを使っても問題ありません。

01-1 グレースケール処理した雲の画像



01-2 レベル補正後



01-3 塗りつぶし後



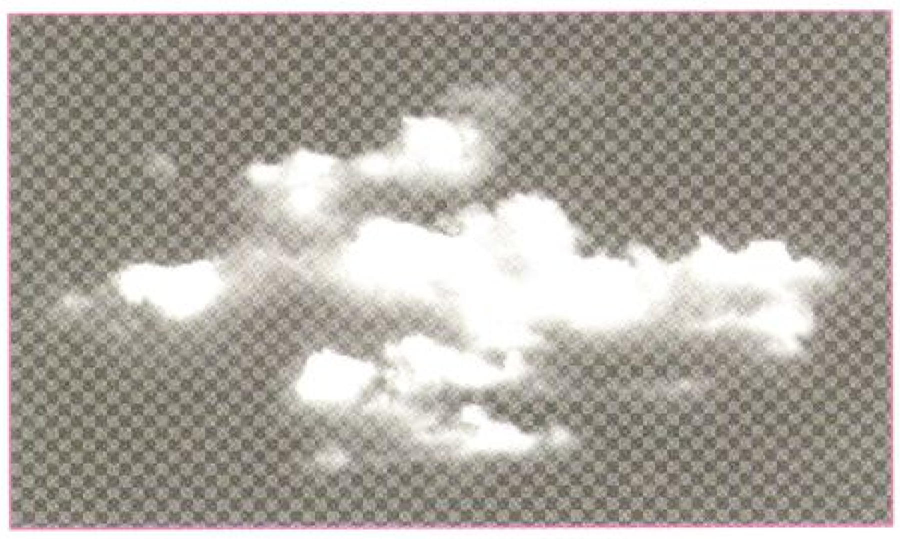
02

チャンネルパネルを開き、「RGB」のチャンネルサムネールをcommand（Ctrl）＋クリックすると、02-1 のように画像の白い部分のみが選択範囲として読み込まれます。レイヤーパネルで新規レイヤーを追加し、「編集」メニュー→“塗りつぶし…”を［使用：ホワイト］、［透明部分の保持：オフ］で実行します。「背景」を削除して、この画像を透過pngとして保存すれば雲のパーツは完成です 02-2。

02-1 白い部分のみが選択される



02-2 透過画像

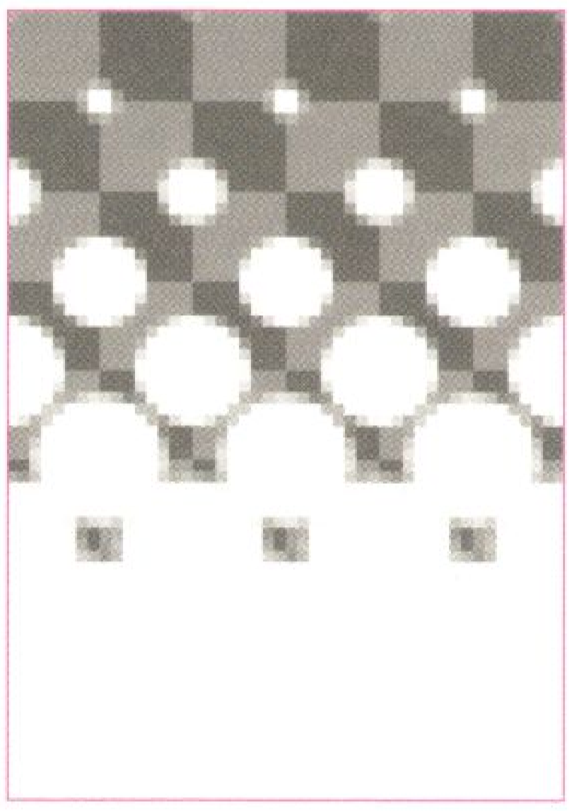


03

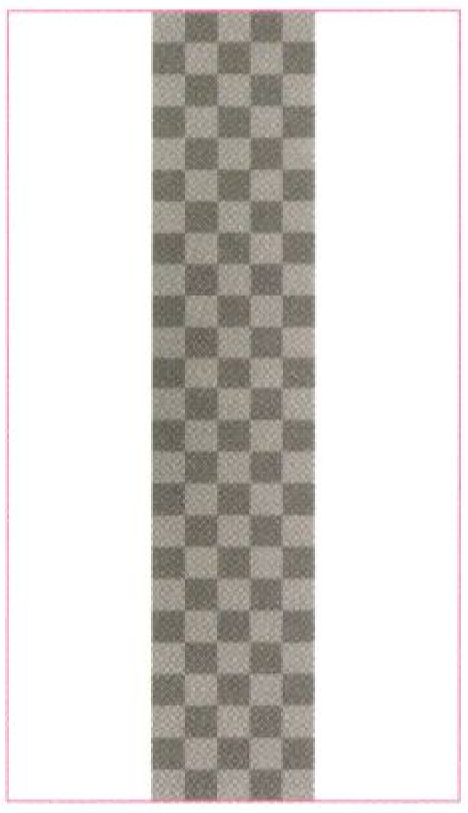
ハーフトーンパターン 03-1、区切り用シャドウ 03-2、新規利用登録の吹き出しボタン 03-3、タイトルロゴ 03-4、枠内背景の縦ストライプ 03-5、ボタンに使う矢印 03-6 の画像も用意しておきます。ハーフトーンのグラデーション、区切り用シャドウは雲と同じ要領で背景透過にしてpng保存しておきます。

MEMO
ハーフトーンパターンは、Photoshopでグラデーションの画像に“ハーフトーンパターン”フィルタを使用して作成します。

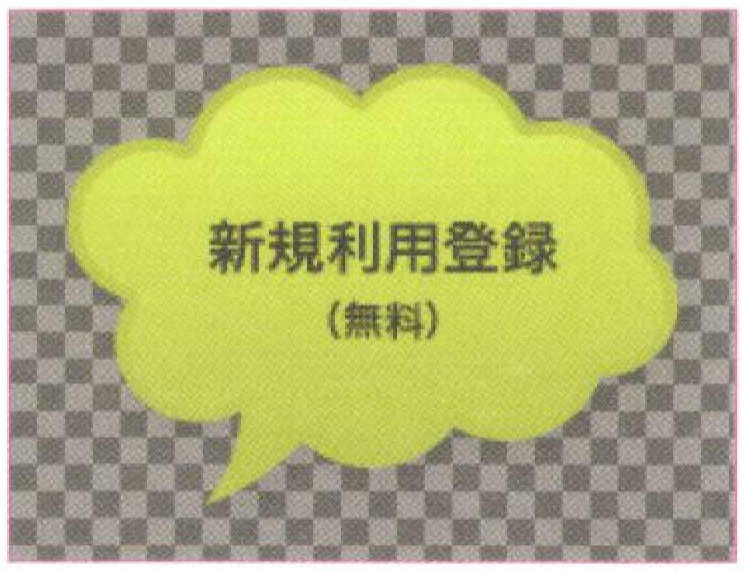
03-1 ハーフトーンパターン



03-2 区切り用シャドウ



03-3 吹き出しボタン



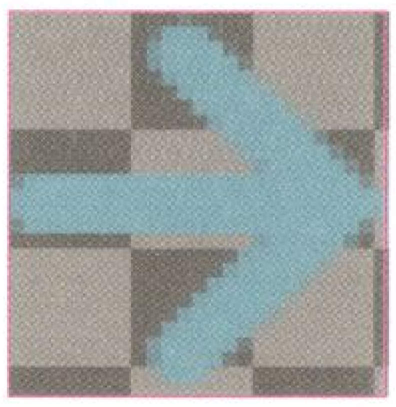
03-4 タイトルロゴ



03-5 縦ストライプ



03-6 矢印



ベースとなるHTML ／ CSSの記述

04

続いてコーディングに入ります。HTMLに 04-1 を記入します。div要素「#content」の中に、タイトル、フォーム要素、新規利用登録ボタン、フッタの要素が入ります。フォームの要素はテキストフィールド、パスワードフィールド、イメージボタンの3種類で、テキストフィールドとパスワードフィールドにはラベルのテキストが付きます。ラベルはlabel要素でマークアップしておきましょう。必要に応じて各要素にクラス名やid名を付加しています。

04-1 ベースとなるHTML

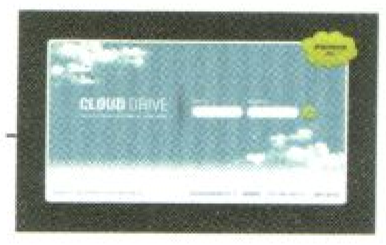
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
— 省略 —
<body>
<div id="content">
  <h1><img src="images/logo.png" width="280" height="67"
  alt="CLOUD DRIVE" /></h1>
  <div id="form">
    <form>
      <p class="userid"><label for="userid">User ID</label><input
      name="userid" type="text" /></p>
      <p class="password"><label for="password">Password</
      label><input name="password" type="password"
      id="password" /></p>
      <p class="login"><input type="image" name="login" id="login"
      src="images/arrow.png" /></p>
    </form>
  </div>
  <div id="regist"><a href="regist.html"><img src="images/register.png"
  width="195" height="150" alt="新規利用登録（無料）" /></a></div>
  <div id="footer">
    <ul>
      <li><a href="about.html">CLOUD DRIVEについて</a></li>
— 省略 —
      <li><a href="inquiry.html">お問い合わせ</a></li>
    </ul>
    <p class="credit">Copyright (C) CLOUD DRIVE co., Ltd. All Rights
    Reserved.</p>
  </div>
</div>
</body>
</html>
```

05

続いてCSSのコーディングです。「style.css」という名前で空のテキストファイルを作成し、HTMLのhead要素内に 05-1 の1行を追加して読み込みます。スタイルはすべてこのファイルに記述していきます。

05-1 HTMLのheadでCSSファイルを読み込む

```
<link href="css/style.css" rel="stylesheet" type="text/css" />
```



06

「style.css」に06-1を記入します。これは、各要素の差異をなくすための初期化処理と、body要素の背景とフォントを指定するものです。

06-1 全体に関わるスタイル（style.css）

```
@charset "UTF-8";

/*各要素の初期設定*/
body, h1, div, p, ul, li, a, form, input, label {
  margin: 0;
  padding: 0;
}
ul, li {
  list-style: none;
}
a {
  text-decoration: none;
  outline: 0;
}
a img {
  border: none;
}
input {
  outline: 0;
}

/*ページ全体のスタイル*/
body {
  background: #353535;
  font-family: "ヒラギノ角ゴ Pro W3",
  "Hiragino Kaku Gothic Pro", "メイリオ",
  Meiryo, Osaka, "ＭＳ Ｐゴシック", "MS
  PGothic", sans-serif;
}
```

個別要素のスタイル作成

07

続けて07-1を追記します。これは全体を内包する「#content」のスタイルです。3～8行目（width:からtop:まで）は、絶対配置とネガティブマージンを使ってブラウザの中央に配置されるように指定しています07-2。

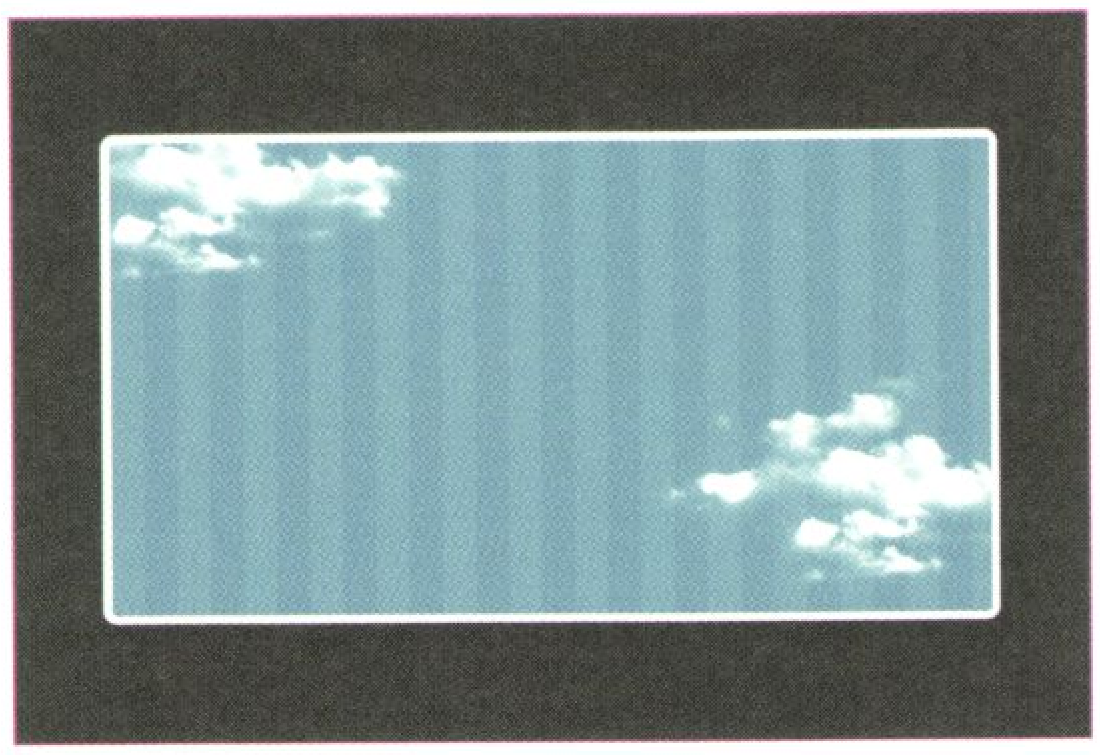
続く9～12行目（backgroud:以下の4行）は背景指定です。カンマ区切りで記述することで、1つの要素に複数の背景を指定することが可能です。ここでは07-3のように、縦ストライプ（bg.png）の上に雲の画像（cloud.png）を2つ載せた状態にしました。ただし、IEではこの指定が無視されてしまうため、CSSハックでIEのみをベースとなるストライプの背景のみに戻し、雲の画像は後ほどJavaScriptを使って挿入します。

22行目は、IEでbox-shadowとborder-radiusを有効にするため「PIE.htc」07-4を読み込んでいます。

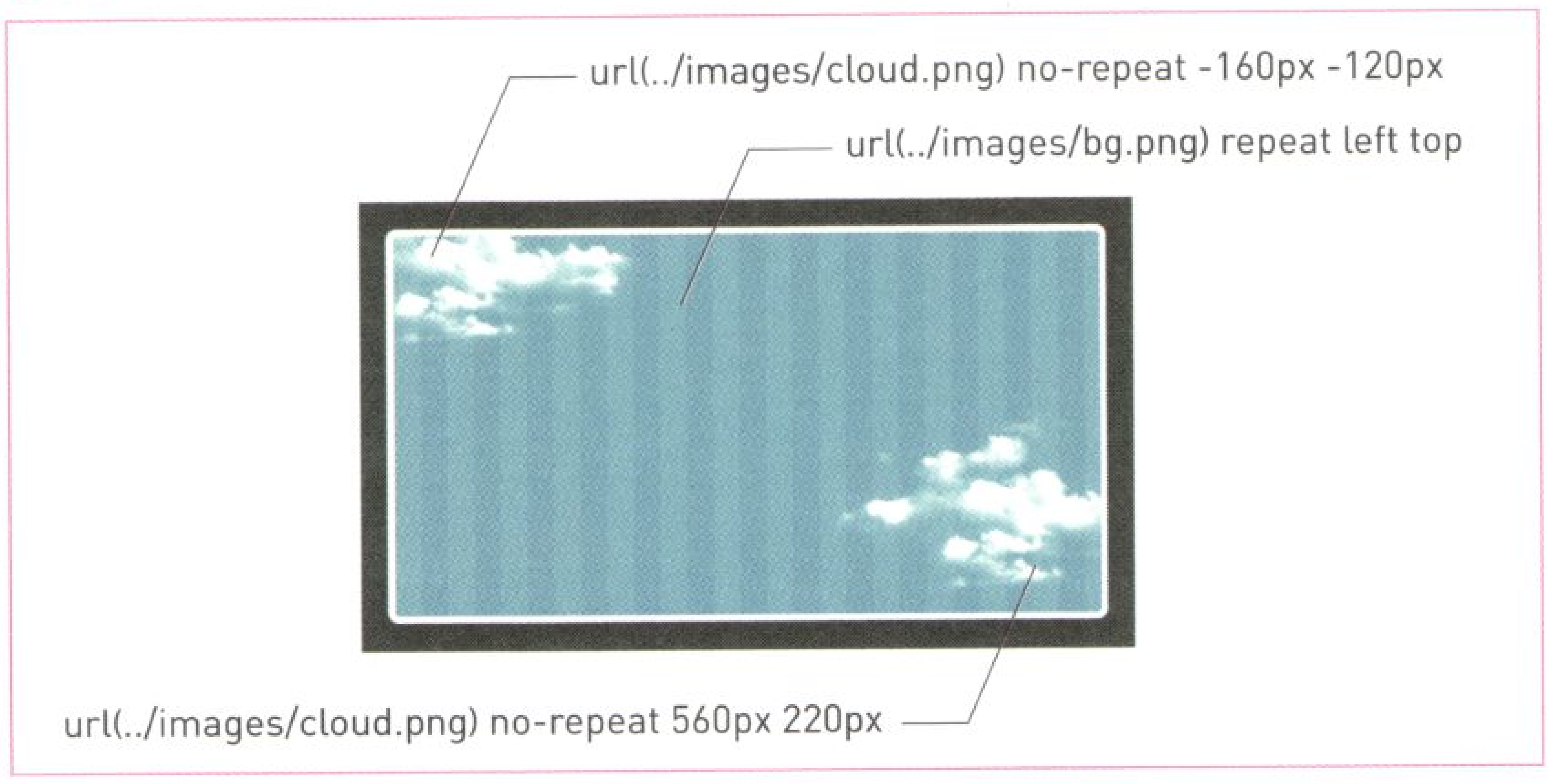
07-1 コンテンツを内包する要素のスタイル（style.css）

```
01 /*コンテンツを内包するdiv要素*/
02 #content {
03   width: 930px;
04   height: 520px;
05   margin: -260px 0 0 -465px;
06   position: absolute;
07   left: 50%;
08   top: 50%;
09   background:
10     url(../images/cloud.png) no-repeat 560px 220px,
11     url(../images/cloud.png) no-repeat -160px -120px,
12     url(../images/bg.png) repeat left top;
13   -webkit-box-shadow: 10px 10px 20px rgba(0, 0, 0, 0.15);
14 ― 省略 ―
15   box-shadow: 10px 10px 20px rgba(0, 0, 0 ,0.15);
16   border: 10px solid #fff;
17   -webkit-border-radius: 15px;
18 ― 省略 ―
19   border-radius: 15px;
20   *background: #2badd0 url(../images/bg.png) repeat left top;        /*IE6,7に適用*/
21   background: #2badd0 url(../images/bg.png) repeat left top¥9;       /*IE8に適用*/
22   behavior: url(../css/PIE.htc); /*動作させるには絶対パスに修正する*/
23 }
```

07-2 「#content」の内容がブラウザの中央に配置される



07-3 縦ストライプの上に雲の画像を2つ載せている



TIPS

複数の背景指定は先に記述したものが上、後に記述したものが下に表示されます。

07-4 PIE.htc

http://css3pie.com/

MEMO

PIE.htcで利用するプロパティには「-pie-」のベンダープレフィックスを付加します（53ページCHAPTER01-04「ウィンドウサイズに連動するリキッドタイプのフリーレイアウト」10のMEMOも参照）。

ATTENTION!

紙面ではPIE.htcを相対パスで指定していますが、実際にはhttpから始まる絶対パスで指定する必要があります。

08

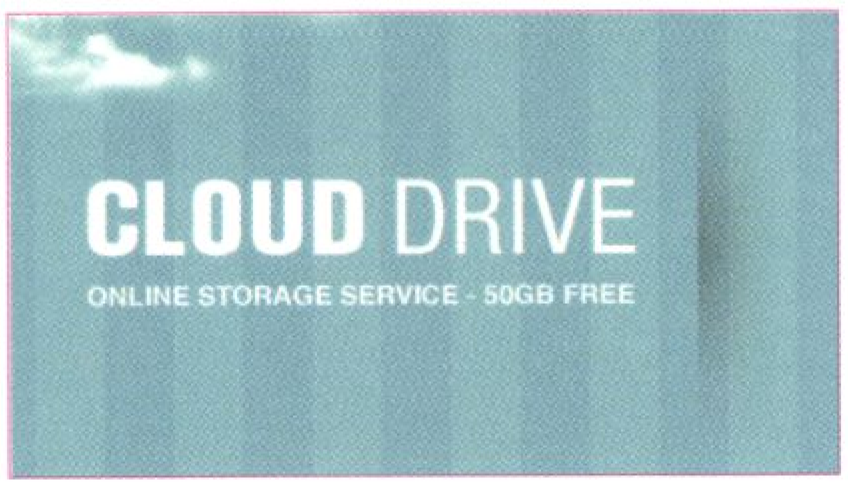
続けて08-1を追記します。「/*タイトル*/」以下は、タイトルのh1要素のスタイルです。h1要素にはタイトルとなるロゴ画像が入っていますが、右に背景画像として区切り用のシャドウを表示します08-2。このように、背景を透過した黒のグラデーションやぼかしは立体的なシャドウとして幅広く利用できます。
「/*フォーム全体を内包するdiv要素*/」以下は、フォーム要素を内包する「#form」の位置を決定します。leftを50%に設定していったん左端を親要素の中央に揃え、マージンを使って誤差を調節します08-3。h1要素も同じ要領で位置を決めています。

08-1 タイトルとフォーム全体を内包するdiv要素のスタイル（style.css）

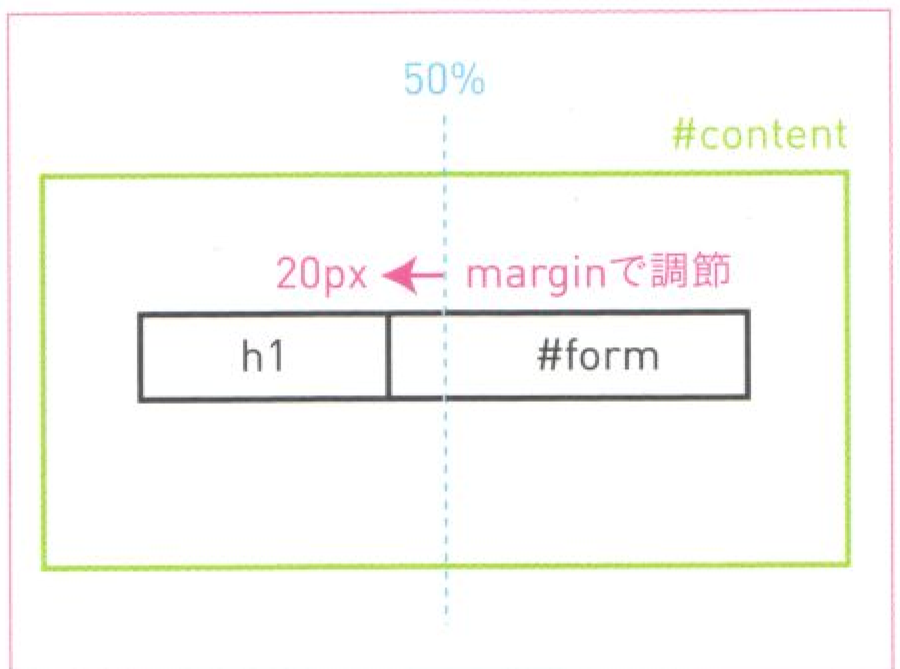
```
/*タイトル*/
#content h1 {
  padding: 70px 70px 70px 0;
  margin: 0 20px 0 0;
  position: absolute;
  right: 50%;
  top: 115px;
  background: url(../images/shadow.png)
  no-repeat right center;
}

/*フォーム全体を内包するdiv要素*/
#form {
  margin: 0 0 0 -20px;
  position: absolute;
  left: 50%;
  top: 185px;
}
```

08-2 タイトルロゴの右に
区切り用のシャドウを表示



08-3 「#form」の位置決定





09

次に 09-1 を追記してフォーム要素やラベルのスタイルを設定します。今回は 09-2 のような、角の丸い個性的な印象のフォームにしました。
通常input要素は、テキストフィールド、ラジオボタン、送信ボタンなど、いろいろな種類のフォーム要素に使われるため、セレクタでの選択が難しくなりますが、属性セレクタを使えば要素の属性値で抽出できるため細かい選択が可能です。
「/*テキストフィールド、パスワードフィールド*/」以下の太字部分ではtype属性にtextとpasswordが設定されているinput要素のみ、「/*イメージボタン*/」以下の太字部分ではimageが設定されているinput要素のみを選択しています 09-3。

09-1 フォーム要素やラベルのスタイル（style.css）

```
/*フォーム要素の配置*/
#form p {
  float: left;
  margin: 0 10px 0 0;
}

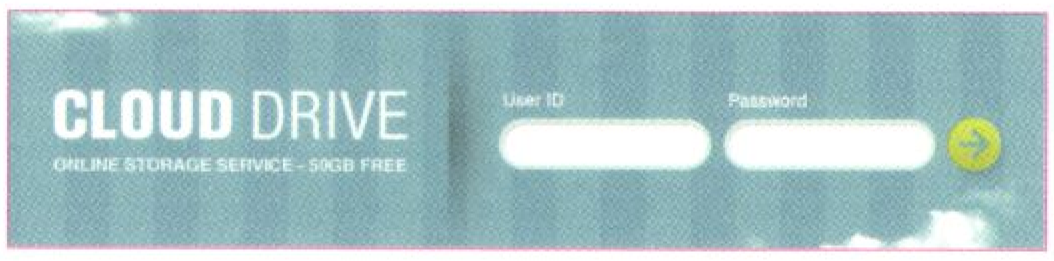
/*テキストフィールドのラベル*/
#form label {
  display: block;
  margin: 0 0 7px 3px;
  font-family: Helvetica, Arial, sans-serif;
  font-size: 14px;
  color: #fff;
}

/*テキストフィールド、パスワードフィールド*/
#form input[type="text"], #form input[type="password"] {
  position:relative;
  width: 140px;
  height: 25px;
  padding: 5px 10px;
  -webkit-box-shadow: inset 1px 2px 1px rgba(0, 0, 0, 0.2);
— 省略 —
  box-shadow: inset 1px 2px 1px rgba(0, 0, 0 ,0.2);
  border: 3px solid #caeff2;
— 省略 —
  border-radius: 20px;
  behavior: url(../css/PIE.htc);     /*動作させるには絶対パスに修正する*/
}
#form input[type="text"]:hover, #form input[type="password"]:hover {
  border-color: #d3da21;
}
#form input[type="text"]:focus, #form input[type="password"]:focus {
  border-color: #0c93c6;
}


/*イメージボタン*/
#form input[type="image"] {
  padding: 8px;
  position: absolute;
  bottom: 0;
  background: -webkit-linear-gradient(#e7ee2a, #c7cd1b);
— 省略 —
  background: linear-gradient(#e7ee2a, #c7cd1b);
  -pie-background: linear-gradient(#e7ee2a, #c7cd1b);
  -webkit-border-radius: 50%;
— 省略 —
  border-radius: 50%;
  -webkit-box-shadow: 3px 3px 5px rgba(0, 0, 0, 0.15);
— 省略 —
  box-shadow: 3px 3px 5px rgba(0, 0, 0 ,0.15);
  behavior: url(../css/PIE.htc);     /*動作させるには絶対パスに修正する*/
}
#form input[type="image"]:hover {
  background: -webkit-linear-gradient(#f9ff60, #ca9d00);
— 省略 —
  background: linear-gradient(#f9ff60, #ca9d00);
  -pie-background: linear-gradient(#f9ff60, #ca9d00);
  behavior: url(../css/PIE.htc);     /*動作させるには絶対パスに修正する*/
}

/*IE9対策*/
#form input[type="image"]:not(:target) {
  left: 350px\9;
  background: #c7cd1b\9;
}
#form input[type="image"]:hover:not(:target) {
  background: #ca9d00\9;
}
```

09-2 角の丸い個性的なフォーム



09-3 属性セレクタを使えば要素の属性値で抽出できる

<input type="text" /> ----▶ input[type="text"]
<input type="password" /> ----▶ input[type="password"]
<input type="image" src="xxx.png" /> ----▶ input[type="image"]

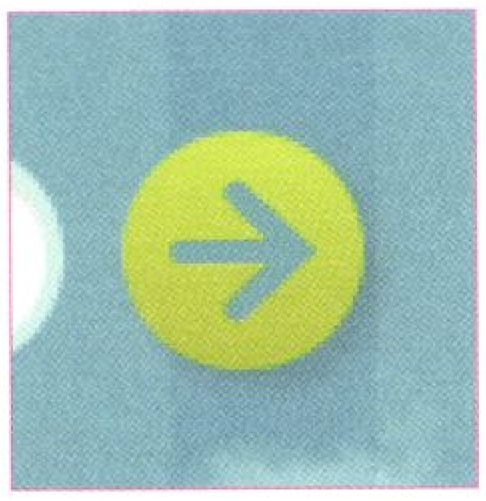
ATTENTION!

IE8以前は属性セレクタに対応していないので、JavaScriptライブラリの「selectivizr」（後述）を利用して同等の機能を実現します。また、IE9は配置とグラデーションに支障が出るので、CSSハックで別処理を施しています。

10

今回の送信ボタンはイメージボタンとしています10-1。画像には背景透過の矢印を使い、CSSで背景をグラデーション、ボタンの形状を円形に設定しました。こうしておけば、色の変更やhover時のスタイルの変化10-2にもCSSのみで柔軟に対応できます。

10-1 送信ボタン



10-2 送信ボタン(hover時)



11

続いて11-1を追記します。新規利用登録の吹き出しボタン11-2を配置するスタイルです。絶対配置で「#content」の右上にレイアウトしますが、この際、11-3のようにrightとheightの値をマイナスに設定することで、枠から飛び出した位置へ配置します。このように、親要素から飛び出す配置をする際には、負の値を設定するテクニックがよく使われるので覚えておきましょう。
hover時には、transformで画像を拡大します。

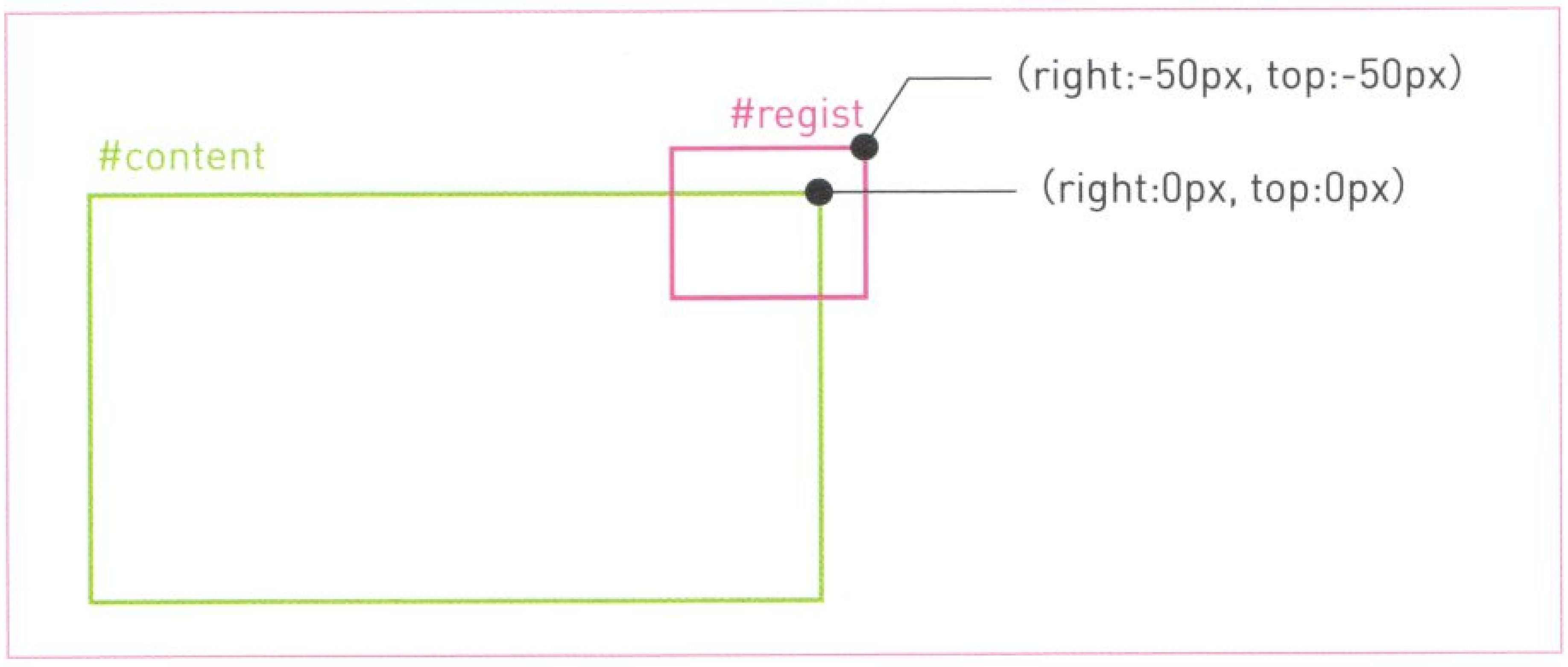
11-1 吹き出しボタンのスタイル (style.css)

```
/*新規利用登録の吹き出し*/
#regist {
  position: absolute;
  right: -50px;
  top: -50px;
}
#regist a:hover img {
  -webkit-transform: scale(1.1, 1.1);
— 省略 —
  transform: scale(1.1, 1.1);
  *zoom: 1.1;   /*IE6,7に適用*/
  zoom: 1.1¥9; /*IE8に適用*/
}
```

11-2 新規利用登録の吹き出しボタン



11-3 枠から飛び出した位置への配置

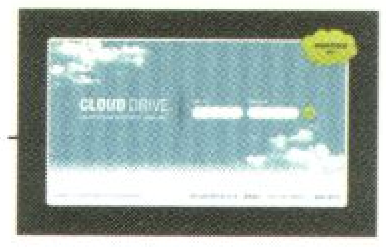


ATTENTION!

IE8以前はtransformに対応していないので、IEの独自プロパティであるzoomを使って拡大します。

12

続いて、フッタのスタイルとなる12-1を追記します。ここでのポイントは、8行目のborder-imageを使ったハーフトーンパターンのボーダーになります。border-imageを使うことで、画像を使ったパターンでボーダーを表現することが可能です。しかも、1枚の画像をスライスする感覚で、上下左右のボーダー、四隅のコーナー、中央の塗りを設定することができます12-2 12-3。border-imageの内容は少し複雑ですが、覚えておいて損はないテクニックといえるでしょう。



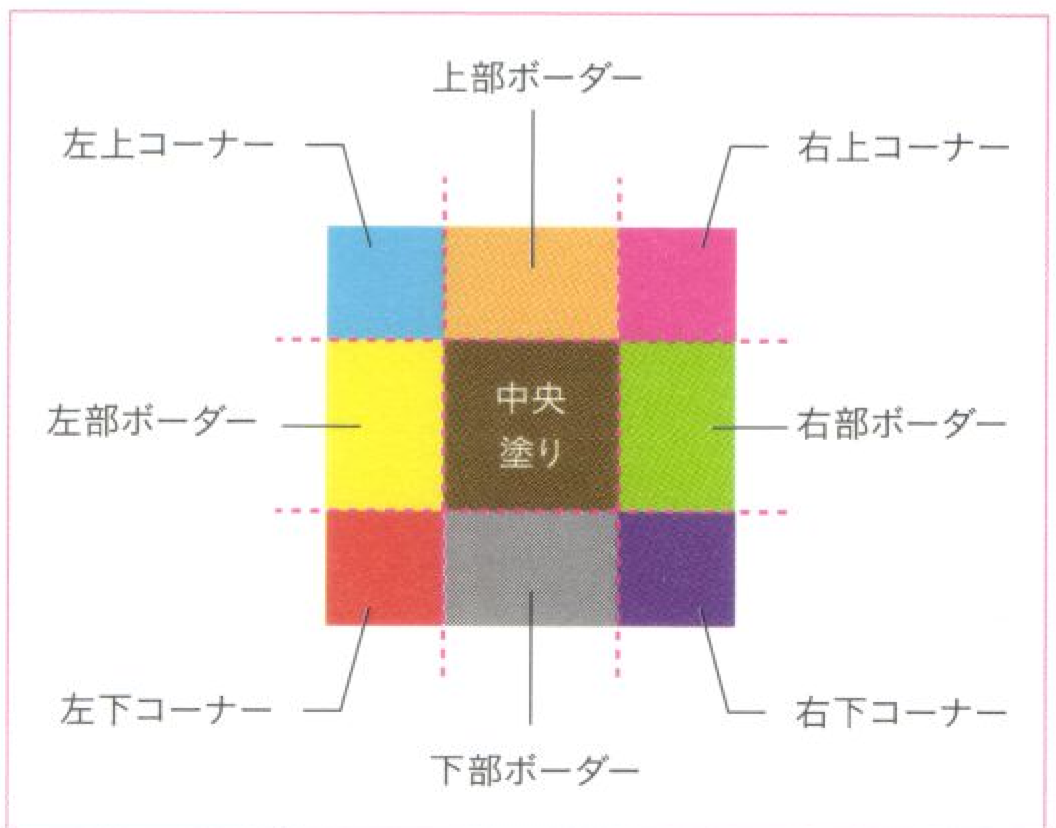
12-1 フッタのスタイル（style.css）

```
/*フッタ*/
#footer {
  width: 100%;
  padding: 60px 0 10px;
  position: absolute;
  left: 0;
  bottom: 0;
  -webkit-border-image: url(../images/
  halftone.png) 50 0 0 0 / 50px 0 0 0 repeat;
— 省略 —
  *background: #fff;   /*IE6,7に適用*/
  background: #fff¥9; /*IE8に適用*/
}

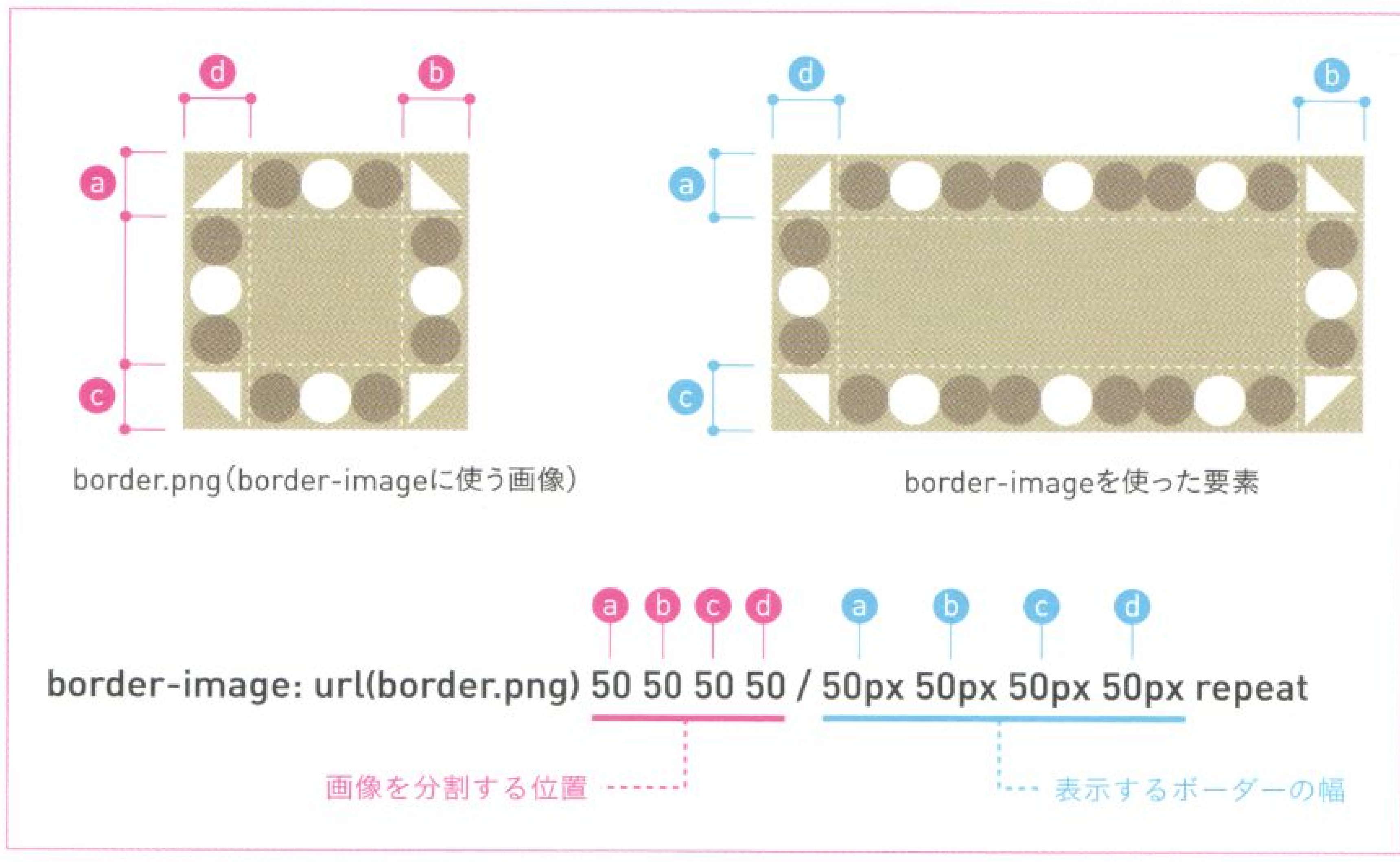
/*フッタ（クレジット）*/
#footer p.credit {
  float: left;
  margin: 5px 0 0 15px;
  font-size: x-small;
  color: #2da6d4;
}

/*フッタ（ナビゲーション）*/
#footer ul {
  float: right;
  margin-right: 15px;
}
#footer ul li {
  display: inline;
  margin: 0 0 0 15px;
}
#footer ul li a {
  font-size: small;
  font-weight: bold;
  color: #2da6d4;
}
#footer ul li a:hover {
  color: #d3da21;
}
```

12-2 1枚の画像を9つのパーツに分割して使う



12-3 border-imageの指定例



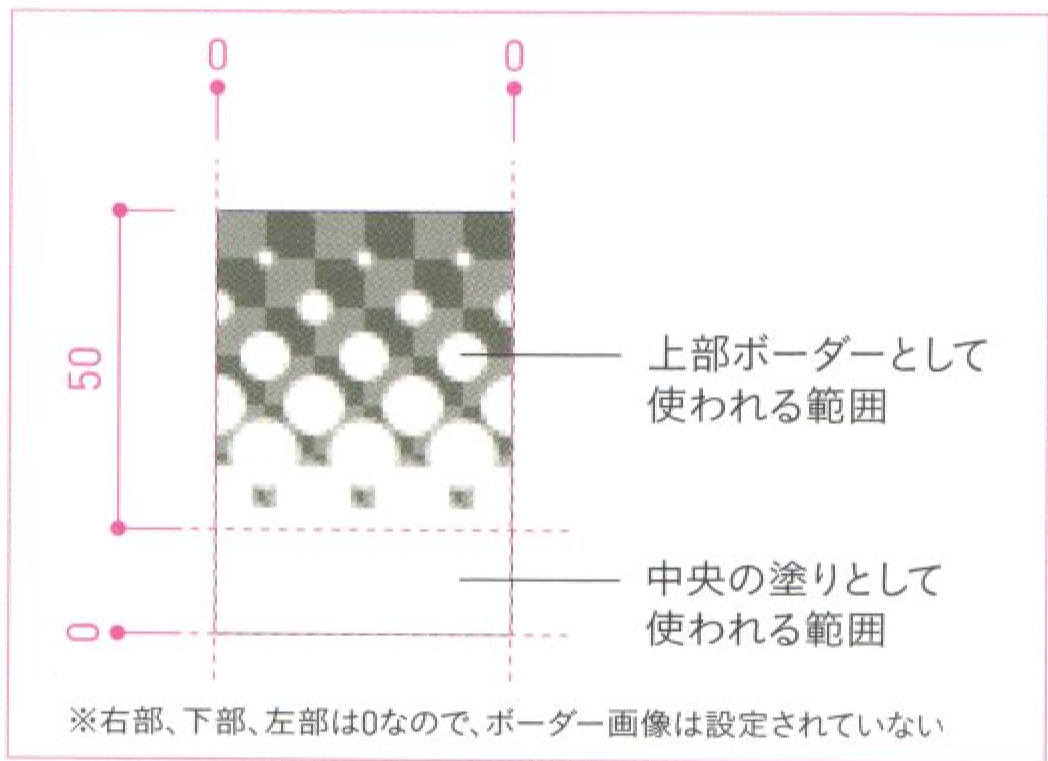
ATTENTION!

IEはborder-imageに対応していないので、JavaScriptを使って対応します（後述）。

13

今回はハーフトーンパターンの画像を **13-1** のようにスライスして使いました。ボーダーの幅は上部のみ指定しているので、他の3辺はボーダーなしの状態になり、結果として上からハーフトーンパターンでグラデーションしているイメージを表現できます **13-2**。

13-1 ハーフトーンパターンの画像をスライスして使用



13-2 上からハーフトーンパターンでグラデーションしているイメージ



IE対策用のスタイル、JavaScriptの作成

14

最後に、IE対策用のJavaScriptで使用するスタイル 14-1 を追記すればCSSは完成です。IEでは、指定できなかった複数背景やborder-imageの代替として空のdiv要素を追加し、そこに画像を表示する処理を行います。

14-1 IE対策JavaScriptで使用するスタイル（style.css）

```
/*IE対策JavaScript用*/
#content .cloudForIE {
  width: 100%;
  height: 100%;
  position: absolute;
  left: 0;
  top: 0;
}
#content .cloud1 {
  background: url(../images/cloud.png) no-repeat -160px -120px;
}
#content .cloud2 {
  background: url(../images/cloud.png) no-repeat 560px 220px;
}
#footer .borderForIE {
  width: 100%;
  height: 50px;
  position: absolute;
  left: 0;
  top: -50px;
  background: url(../images/halftone.png) repeat-x left top;
}
```

15

HTMLに戻り、「style.css」を読み込んでいるlink要素の前に 15-1 を挿入します。IE7と8では、左上の雲画像を表示するためのdiv要素2つとフッタのハーフトーンパターンを表示するためのdiv要素を追加します。IE9は複数背景に対応しているので、フッタのハーフトーンパターン用div要素のみです。なお、今回の作例では解説が複雑になるためIE6をサポートしていません 15-2 15-3。必要があればIE6用の振り分けを行い、メッセージを表示するなどの処理を検討しましょう。

15-1 HTMLのheadでJavaScriptを読み込む

```
<!--[if (gte IE 7)&(lte IE 9)]>
  <script type="text/javascript" src="js/jquery-1.7.1.js"></script>
  <script type="text/javascript" src="js/selectivizr.js"></script>
<![endif]-->
<!--[if (gte IE 7)&(lte IE 8)]>
  <script type='text/javascript'>
    $(document).ready(function() {
      $('#content')
        .prepend('<div class="cloudForIE cloud1"></div>')
        .prepend('<div class="cloudForIE cloud2"></div>');
      $('#footer').prepend('<div class="borderForIE"></div>');
    });
  </script>
<![endif]-->
<!--[if IE 9]>
  <script type='text/javascript'>
    $(document).ready(function() {
      $('#footer').prepend('<div class="borderForIE"></div>');
    });
  </script>
<![endif]-->
```



15-2 作例の完成イメージ（正常な表示結果）



15-3 IE6での表示結果



MEMO

15-1で読み込んでいる「selectivizr.js」01はIE8以前でサポートされていないCSSセレクタを使えるようにするものです。jQuery 02などのJavaScriptライブラリと合わせて使います。

01 selectivizr.js

http://selectivizr.com/

02 jQuery

http://jquery.com/

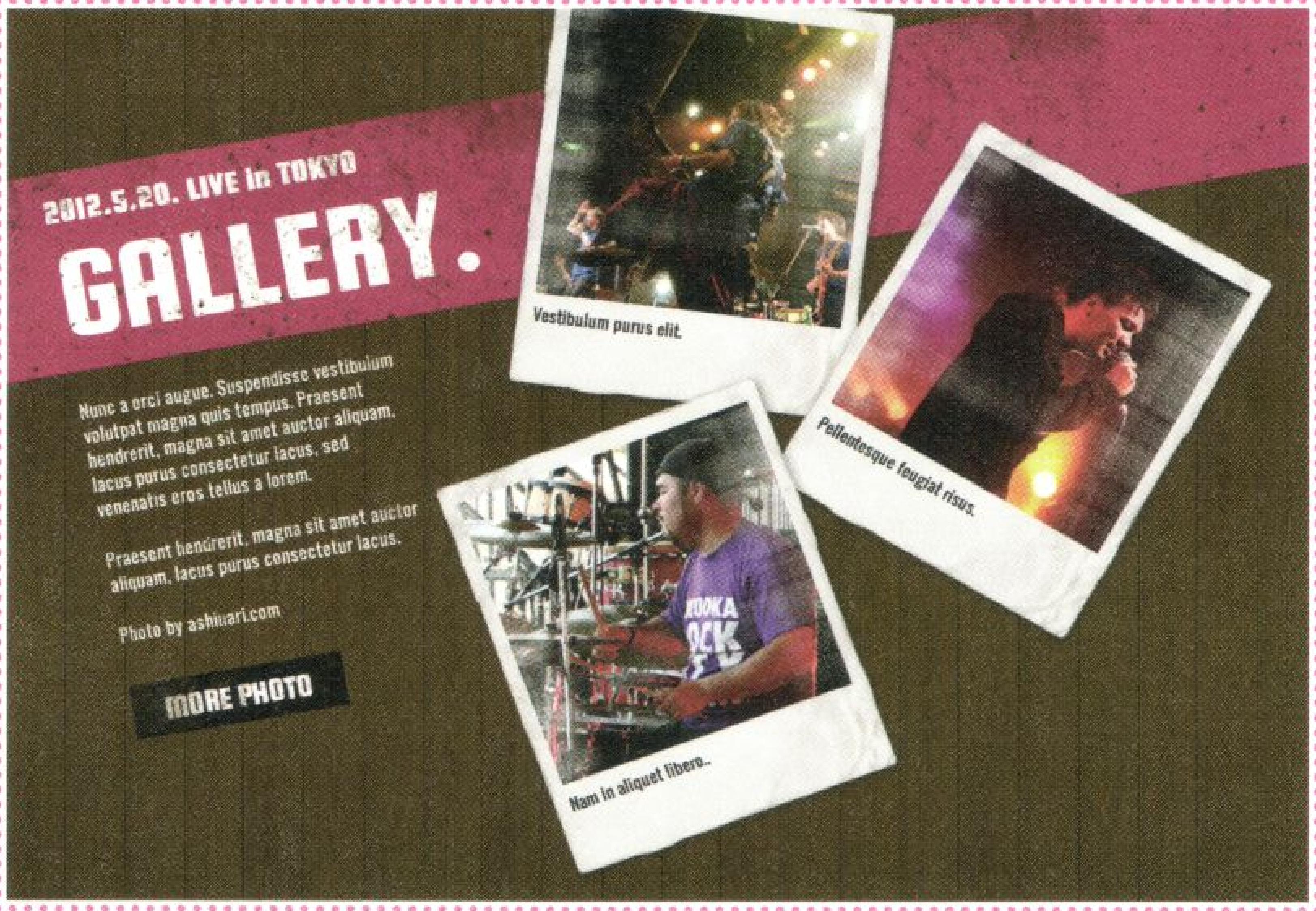
ATTENTION!

「selectivizr」はローカルでは動作しないことがあるので、サーバーにアップロードしてからの確認が必要です。また、外部CSSしか認識しないため、埋め込みやインラインでの記述では動作しません。

まとめ [CONCLUSION]

少し前まで透過画像はgifのみでした。gifでは半透明の処理ができず、デザインパーツとして利用する場合に多くの制約に悩まされましたが、モダンブラウザの普及した近年はpngの透過画像が比較的自由に使えるようになり、画像による表現力が大きく広がっています。単純なドロップシャドウやグラデーションなどはCSS3のみでも表現できるようになってきましたが、まだまだ複雑なものは画像に頼らざるを得ません。状況に応じてうまく使い分け、効果的なデザインを目指しましょう。

07 CSS3と画像を組み合わせたアナログテイストのデザイン



写真枠のパーツや掠れなどのテクスチャを使ったアナログテイストのデザインを実現します。WebKitベースのSafariやChromeでは文字をテクスチャでマスキングして、質感のある印象に仕上げます。要素を回転したり、写真に掠れのテクスチャを重ねて、よりアナログな印象を高めましょう。

制作・文／高橋としゆき（Graphic Arts Unit）
写真提供／写真素材 足成
http://www.ashinari.com/

制作のポイント

- テキストにWebフォントを利用する
- mask-imageでテキストをマスキング
- rotateを使って要素を傾ける

対応ブラウザ

IE	P.E.	Safari	5 以上
Firefox	3 以上	Chrome	12 以上

※IE6以上で表示は確保されるが、一部の処理に非対応。

TECHNIC 1

テキストや写真をテクスチャでマスキングする

写真から作ったテクスチャでテキストをマスキングすることで、アナログ感を強調します（画像上）。また、写真の上から掠れのテクスチャを重ねることで、リアルな表現を演出します（画像下）。



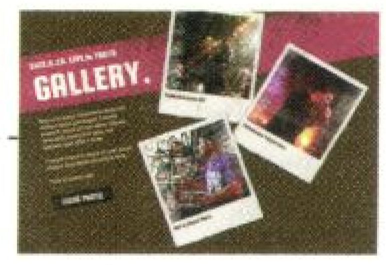


TECHNIC 2

要素をランダムに配置する

タイトルやサムネール写真などの要素をランダムに傾け、レイアウトに動きを出します。



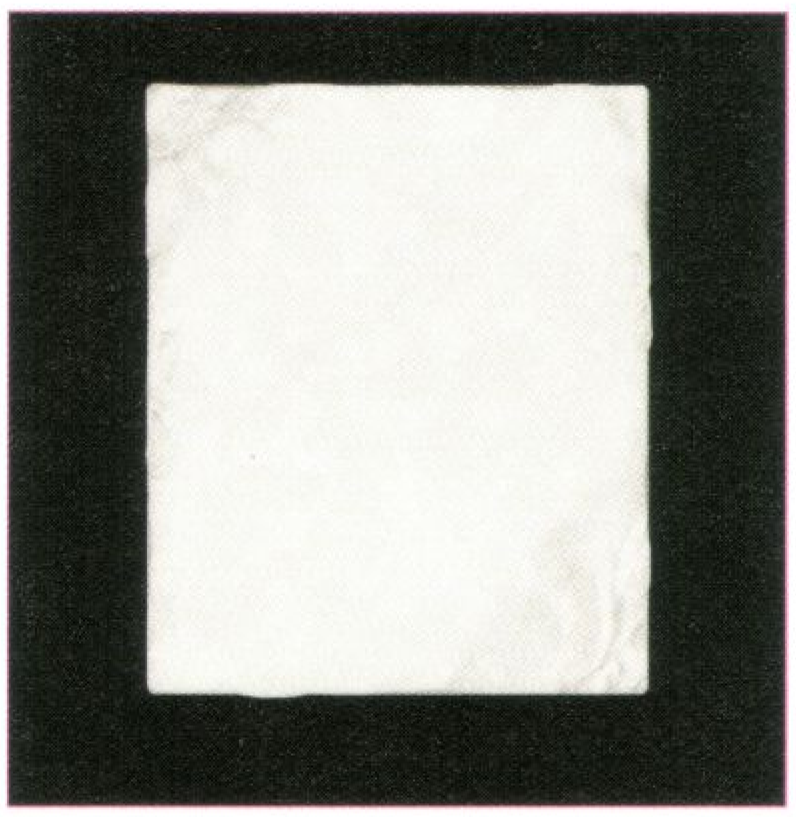


パーツを準備する

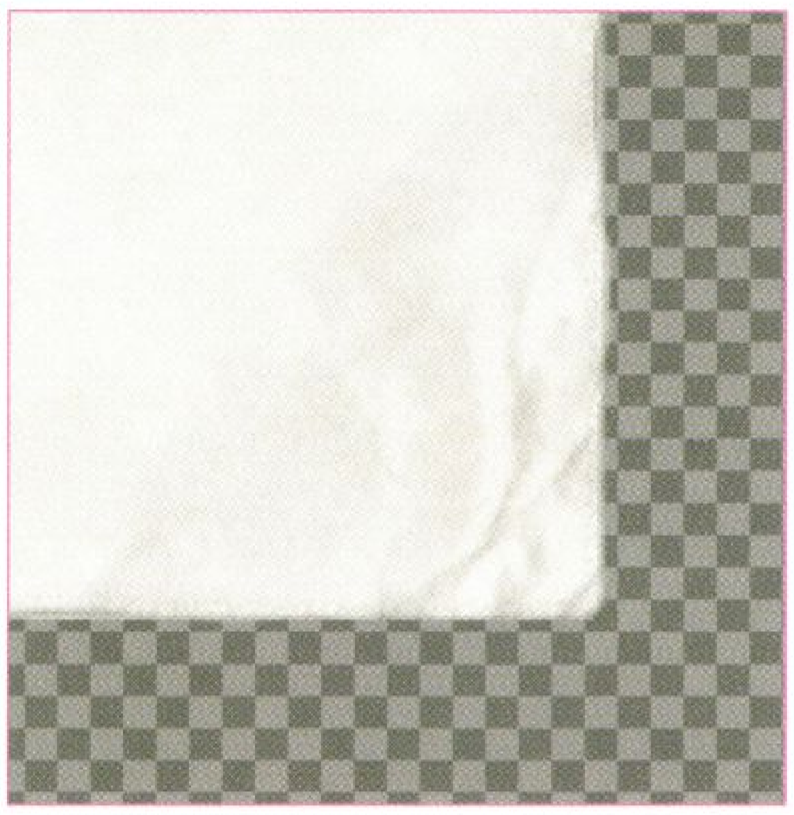
01

まずはレイアウトに使うパーツを用意します。最初は写真枠です。01-1のような紙のテクスチャを撮影、またはスキャンし、切り抜きます 01-2。Photoshopのレイヤー効果でドロップシャドウを加えたら、透過pngとして保存しましょう 01-3 01-4。さらに、背景に使う木目のテクスチャ 01-5と、写真画像3点 01-6も用意しておきます。

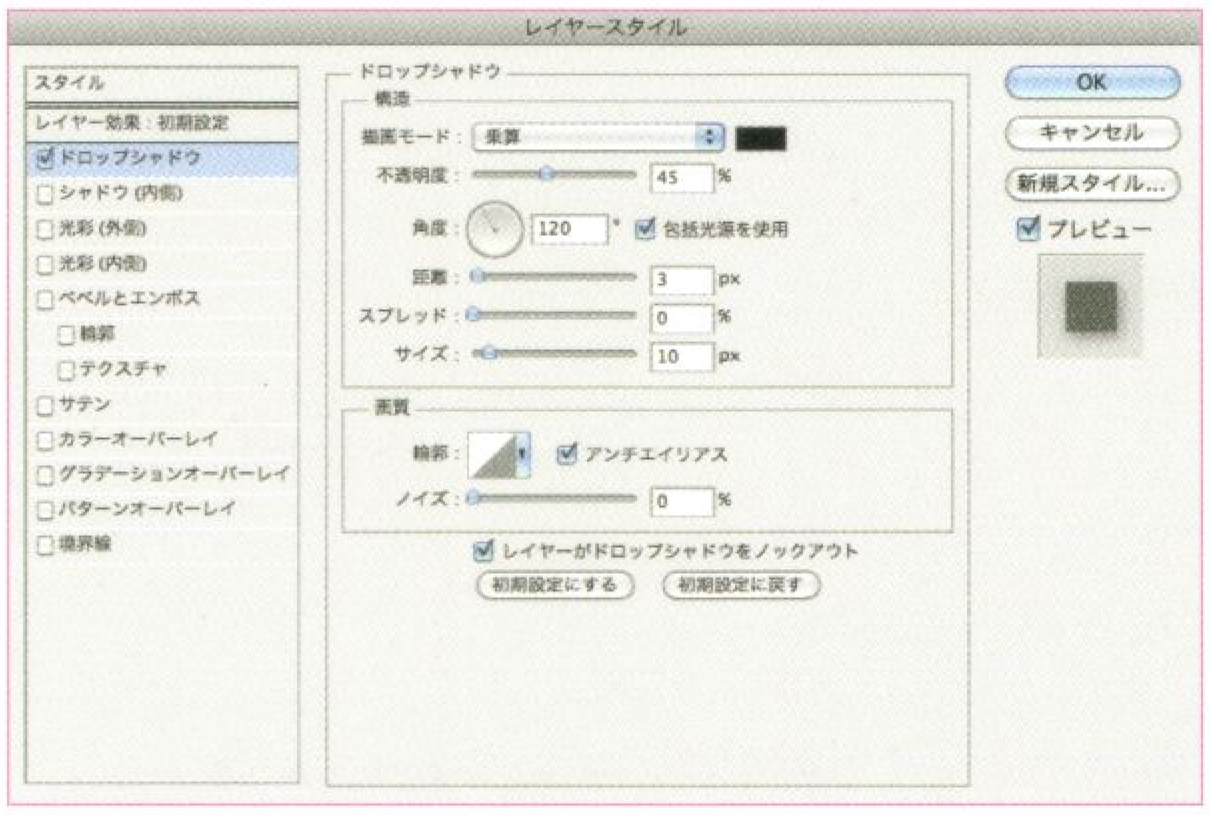
01-1 紙のテクスチャ



01-2 背景を透明にする



01-3 ドロップシャドウの設定画面



01-4 ドロップシャドウを追加



01-5 木目のテクスチャ



01-6 写真3点

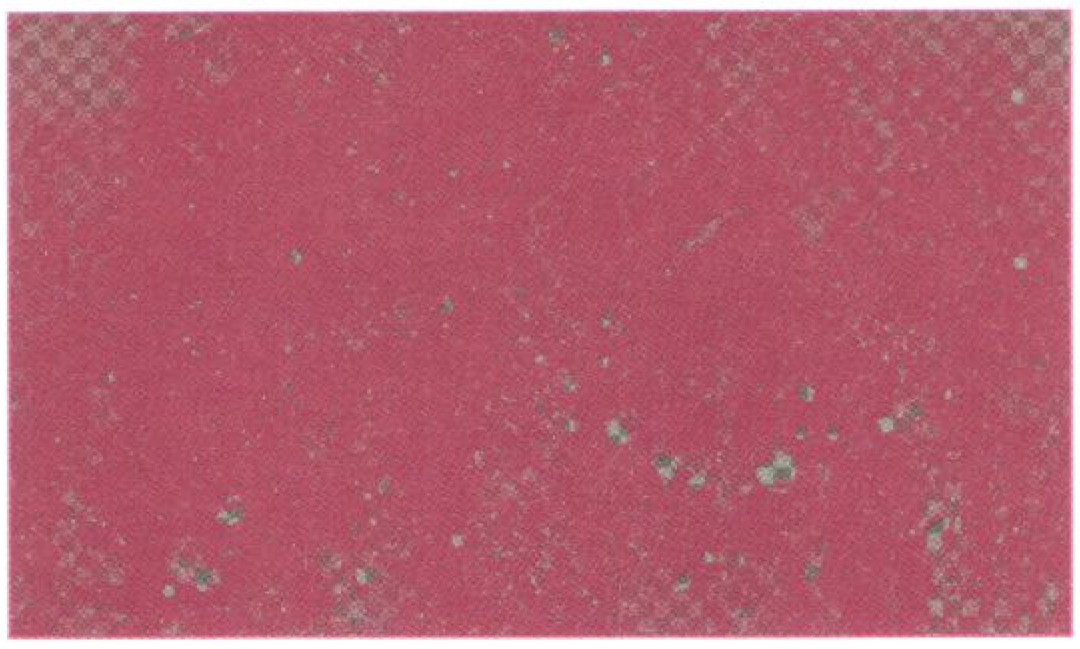


> **MEMO**
>
> ここではMac OS X版のPhotoshop CS5を使用していますが、画像を同じような状態で保存できれば、Fireworksなど、ほかのデザインツールや画像処理ソフトを使っても問題ありません。

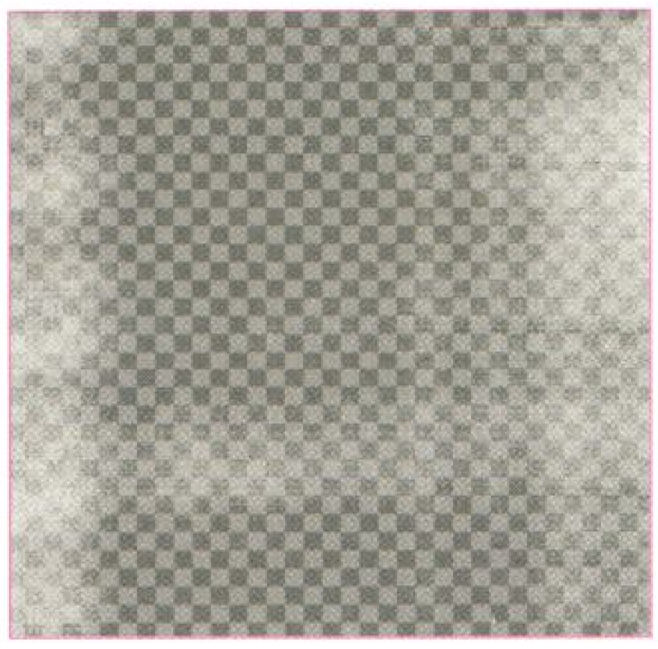
02

続いて、タイトル部分などのマスクとして使うテクスチャを用意します。これにはコンクリートの写真を利用して作った透過画像（mask.png）02-1を用います。さらに同じ要領で、写真に掠れた効果を付けるための画像も用意します。今回はコピー機で作ったノイズ（noise.png）02-2を利用しました。両方を透過pngとして保存します。

02-1 コンクリートの写真を利用して作った透過画像



02-2 コピー機で作ったノイズ



ベースとなるHTML／CSSの作成

03

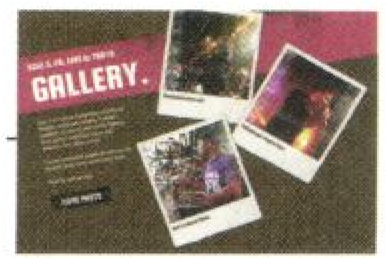
続いてHTMLのコーディングです。HTMLを03-1のように記述します。タイトル等の入るdiv要素「#header」、写真を配置するためのul要素「#thumb」があり、これら全体を「#content」というidのdiv要素で囲みます。

「#header」の最後には、詳細ページへ移動するためのボタンがあります。これは、p要素「.btn」の中に「MORE PHOTO」というテキストのリンクを入れた形です。

「#thumb」のli要素は3つあるので、写真は3点ということになります。それぞれのli要素には写真とキャプションが入っており、1つずつ個別のクラス名を付加しておきます。ここでは「.thumb1」「.thumb2」「.thumb3」としました。

03-1 ベースとなるHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
— 省略 —
<body>
<div id="content">
  <div id="header">
    <h1><span>2012.5.20. LIVE in TOKYO</span>GALLERY.</h1>
    <p>Nunc a orci augue. Suspendisse vestibulum volutpat magna quis tempus. Praesent
    hendrerit, magna sit amet auctor aliquam, lacus purus consectetur lacus, sed
    venenatis eros tellus a lorem.</p>
    <p>Praesent hendrerit, magna sit amet auctor aliquam, lacus purus consectetur
    lacus.</p>
    <p>Photo by ashinari.com</p>
    <p class="btn"><a href="#">MORE PHOTO</a></p>
  </div>
  <ul id="thumb">
    <li class="thumb1">
      <div class="photo"><img src="images/photo1.jpg" width="280" height="280"
      alt="photo" /></div>
      <p class="caption"><a href="#">Vestibulum purus elit.</a></p>
    </li>
    <li class="thumb2">
      <div class="photo"><img src="images/photo2.jpg" width="280" height="280"
      alt="photo" /></div>
      <p class="caption"><a href="#">Pellentesque feugiat risus.</a></p>
    </li>
    <li class="thumb3">
      <div class="photo"><img src="images/photo3.jpg" width="280" height="280"
      alt="photo" /></div>
      <p class="caption"><a href="#">Nam in aliquet libero..</a></p>
    </li>
  </ul>
</div>
</body>
</html>
```



04

続いてCSSのコーディングに移りますが、IEだけは個別の処理をしてやる必要があるので、対応用に別のCSSファイルを用意します。基本のスタイルを「style.css」、IE6～8用のスタイルを「styleforie.css」という名前にします。それぞれのファイル名で空のテキストファイルを2つ用意し、HTMLのhead要素に04-1を追記します。

2～4行目は、IE6で透過pngを扱うためのJavaScriptプラグイン「DD_belatedPNG」04-2を利用するための記述です。続く5行目以降でIE6～8を対象に「styleforie.css」を読み込みます。

04-1 HTMLのheadでJavaScriptとCSSを読み込む

```
01 <link rel="stylesheet" type="text/css" href="css/style.css" />
02 <!--[if IE 6]>
03   <script type="text/javascript" src="js/DD_belatedPNG_0-1.0.8a.js"></script>
04 <![endif]-->
05 <!--[if (gte IE 6)&(lte IE 8)]>
06   <link rel="stylesheet" type="text/css" href="css/styleforie.css" />
07 <![endif]-->
```

04-2 DD_belatedPNG

http://www.dillerdesign.com/experiment/DD_belatedPNG/

05

「style.css」に05-1を記入します。「/*Google web fontsの読み込み*/」以下では、Google Web Fonts 05-2から必要なフォントを読み込むよう指定します。

そして「/*各要素の初期設定*/」以下では各要素のmarginとpaddingを0に設定し、「/*ページ全体のスタイル*/」以下でページ全体の背景画像を指定します。

「/*コンテンツ全体のスタイル*/」以下では要素全体を囲む「#content」、その下ではタイトルとリードコピー、「MORE PHOTO」ボタンを含む「#header」のスタイルを設定します。「#header」はtransformで反時計回りに10°回転させます05-3。

05-1 全体に関わるスタイル（style.css）

```
@charset "UTF-8";

/*Google Web Fontsの読み込み*/
@import url(http://fonts.googleapis.com/css?family=Oswald|Squada+One);

/*各要素の初期設定*/
body, h1, p, span {
  margin: 0;
  padding: 0;
}

/*ページ全体のスタイル*/
body {
  background: url(../images/bg.jpg) repeat;
}

/*コンテンツ全体のスタイル*/
#content {
  width: 100%;
  height: 820px;
  overflow: hidden;
}

/*タイトル、リードコピー、MORE PHOTOリンクを内包するdiv要素のスタイル*/
#header {
  width: 2500px;
  margin: -50px 0 0 -25px;
  -webkit-transform: rotate(-10deg);
― 省略 ―
  transform: rotate(-10deg);
}
```

05-2 Google Web Fonts

http://www.google.com/webfonts/

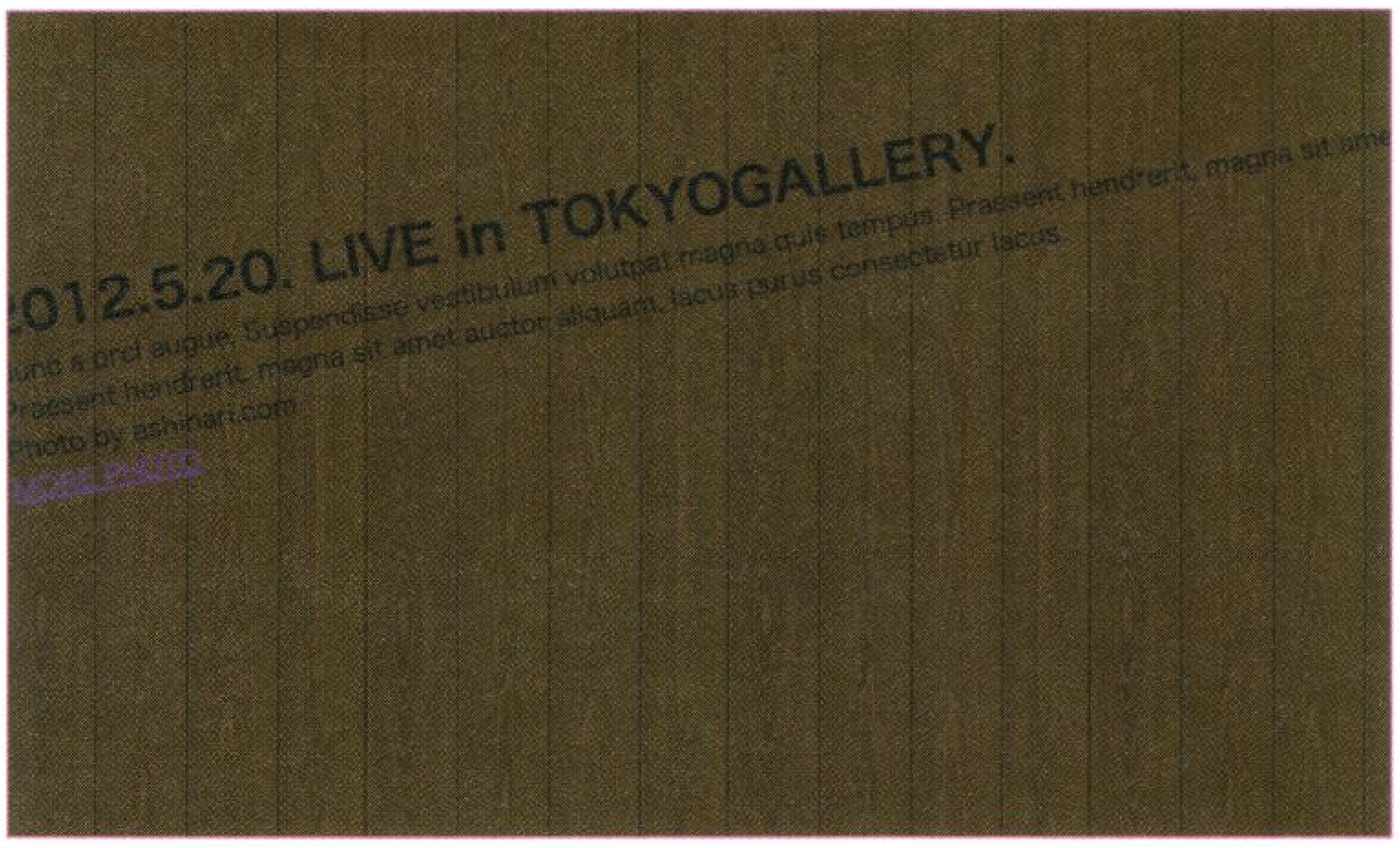
ATTENTION!

IEでは、ローカル上で確認するとフォントが正常に表示されない場合があります。

ATTENTION!

IEはtransformに対応していないので回転させることはできませんが、表示内容の可読性は保持できます。

05-3 「#header」を反時計回りに10°回転させている



MEMO

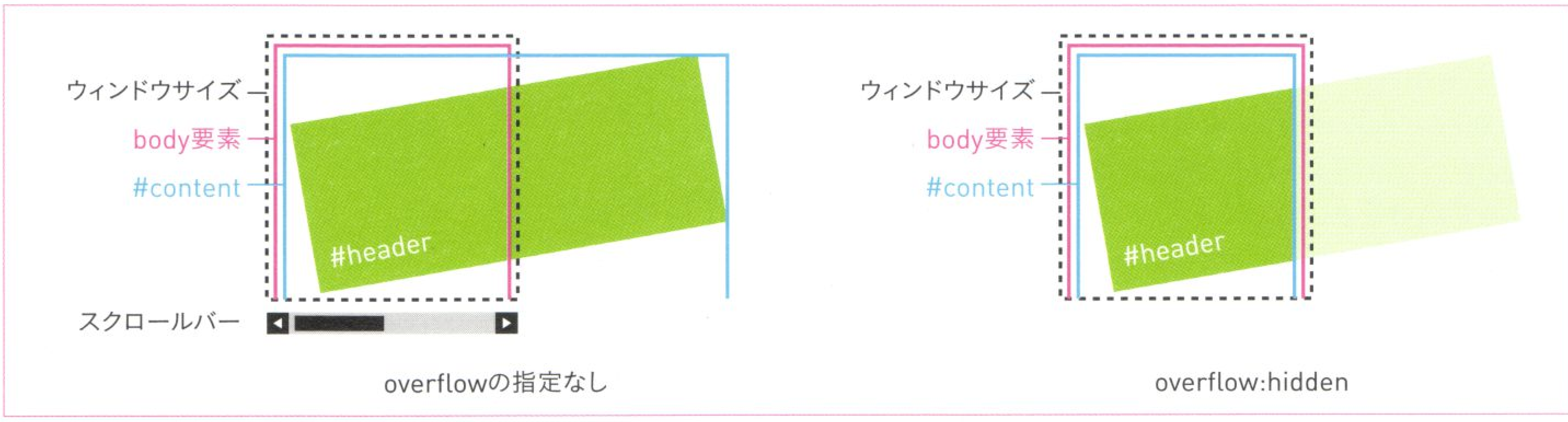
transformプロパティを使うと、いろいろな変形を行うことができますが、合わせてtransform-originプロパティを用いれば、変形の基準点を要素に対するX座標とY座標で指定できます。例えば「transform-origin: 0 0」のように指定すると、要素の左上が変形の基準となります。今回はtransform-originプロパティを使っていないので、要素の中央が基準点になっています。

タイトル帯などのスタイルの作成

06

今回はh1要素を帯状のデザインにしますが、「#header」の角度を右上がりに回転させたときにh1の帯を画面外へはみ出すところまで伸ばしたいと思います。単純に考えればh1要素のwidthを長めに設定すれば問題ありませんが、そのままだと横方向のスクロールバーが出てしまいます。それを考慮して、「#header」を内包する「#content」のwidthを100%にした上で、overflowをhiddenに設定します 06-1。こうすれば、画面サイズを縮めてもスクロールバーが表示されることがありません。「#header」の細かい位置はmarginを使って微調節します。

06-1 スクロールバーを出さないための工夫



MEMO

写真のul要素「#thumb」も「#content」の内部にありますが、最終的にpositionをabsoluteに設定して通常フローから外すため、「#content」によって隠れてしまうことはありません。



07

続けて 07-1 を追記します。「/*タイトル帯のスタイル*/」以下はタイトルのh1要素のスタイルです。帯の背景にマスク用画像を表示しますが、「/*webkitのみ上書き*/」以下でWebKitベースのブラウザ（SafariやChrome）のみを対象にbackgroundを書き換え、さらにmask-imageで要素全体をマスク画像でマスキングします 07-2 07-3。こうすることでテキストにもテクスチャが加わり、アナログなイメージになります。その下のリードコピー、「MORE PHOTO」ボタンのスタイルでもh1と同様にmask-imageを利用しています。

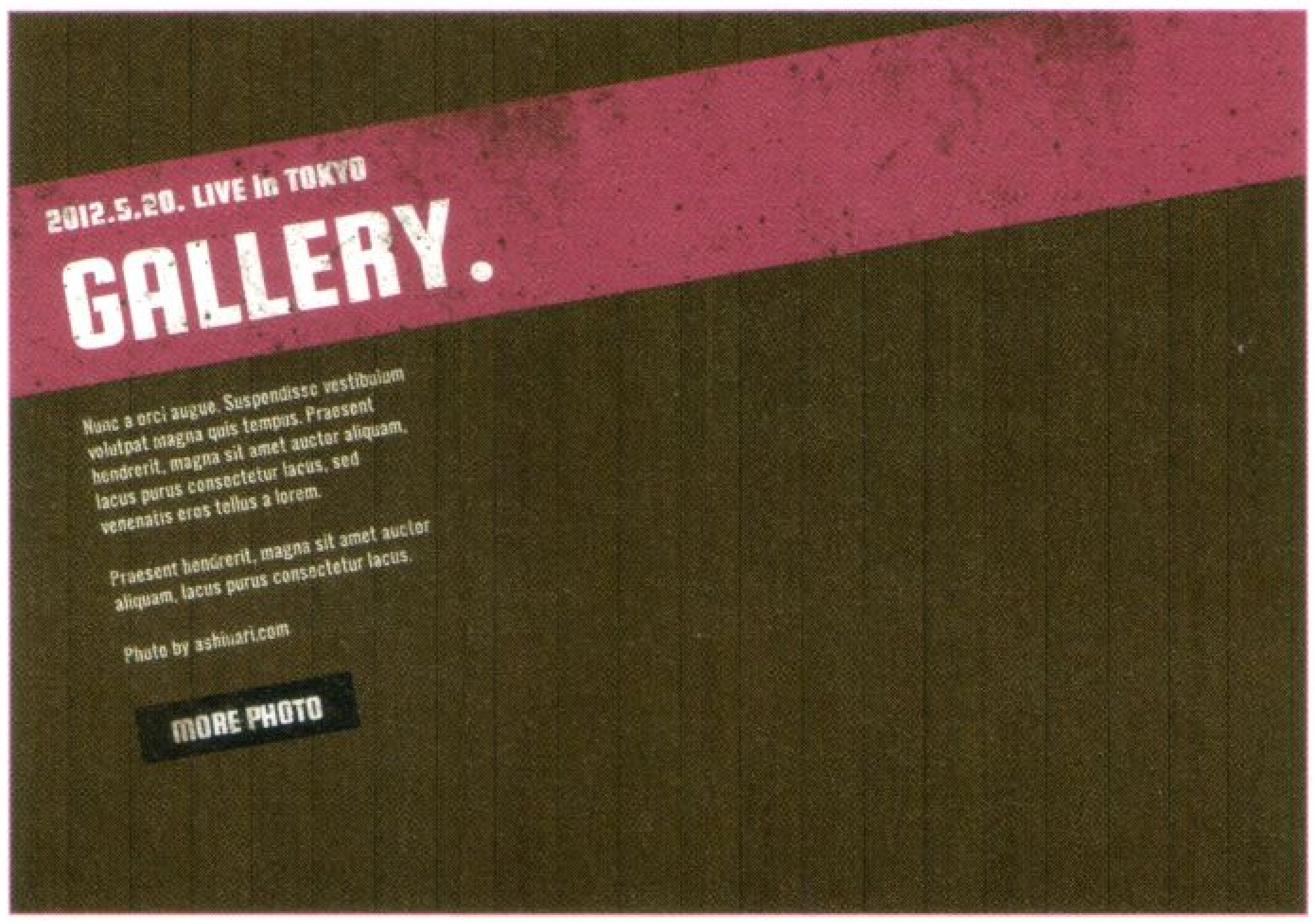
07-1 タイトル帯などのスタイル（style.css）

```
/*タイトル帯のスタイル*/
#header h1 {
  font-size: 120px;
  color: #fff;
  font-family: 'Squada One', cursive;
  padding: 20px 0 10px 80px;
  position: relative;
  background: url(../images/mask.png);
}
/*webkitのみ上書き*/
@media screen and (-webkit-min-device-pixel-ratio:0){
  #header h1{
    background-image: none;
    background: #c90050;
    -webkit-mask-image: url(../images/mask.png);
    mask-image: url(../images/mask.png);
  }
}
#header h1 span {
  display: block;
  font-size: 30px;
}

/*リードコピーのスタイル*/
#header p {
  width: 300px;
  padding: 25px 0 0 80px;
  font-family: 'Oswald', sans-serif;
  color: #fff;
  -webkit-mask-image: url(../images/mask.png);
  mask-image: url(../images/mask.png);
}

/*MORE PHOTOリンクのスタイル*/
p.btn a {
  display: inline-block;
  padding: 10px 30px;
  margin-top: 12px;
  background: #000;
  font-family: 'Squada One', cursive;
  color: #fff;
  text-decoration: none;
  font-size: 30px;
  -webkit-mask-image: url(../images/mask.png);
  mask-image: url(../images/mask.png);
}
p.btn a:hover {
  background: #c90050;
  color: #000;
}
```

07-2 テキストのマスキング結果



07-3 マスキングの仕組み



MEMO

mask-imageは、指定した画像のアルファチャンネル（透明度）に従って要素をマスキングするプロパティです。また、マスク用のソースには画像だけでなくグラデーションも利用できます。

ATTENTION!

現在mask-imageをサポートしているのはWebKitのみなので、それ以外のブラウザではテキストにマスクは適用されませんが、可視性に支障はありません。

写真のスタイルの作成

08

続いて、写真に関するスタイルとして 08-1 を追記します。まずは写真3点ともに適用する共通スタイルで、li要素の背景に写真枠の画像を使い、positionにabsoluteを指定して、要素を絶対配置にします。

それ以降では、写真3点のスタイルを個別に指定しています。要素の位置をleft、topのプロパティで指定します。これは親要素の左上を原点とした座標です。

さらに、「#header」と同様にtransform: rotateを使って写真の角度を指定します。こうすることでランダムな配置がより強調され、写真をばらまいたようなイメージに仕上がります 08-2。

08-1 写真枠のスタイル（style.css）

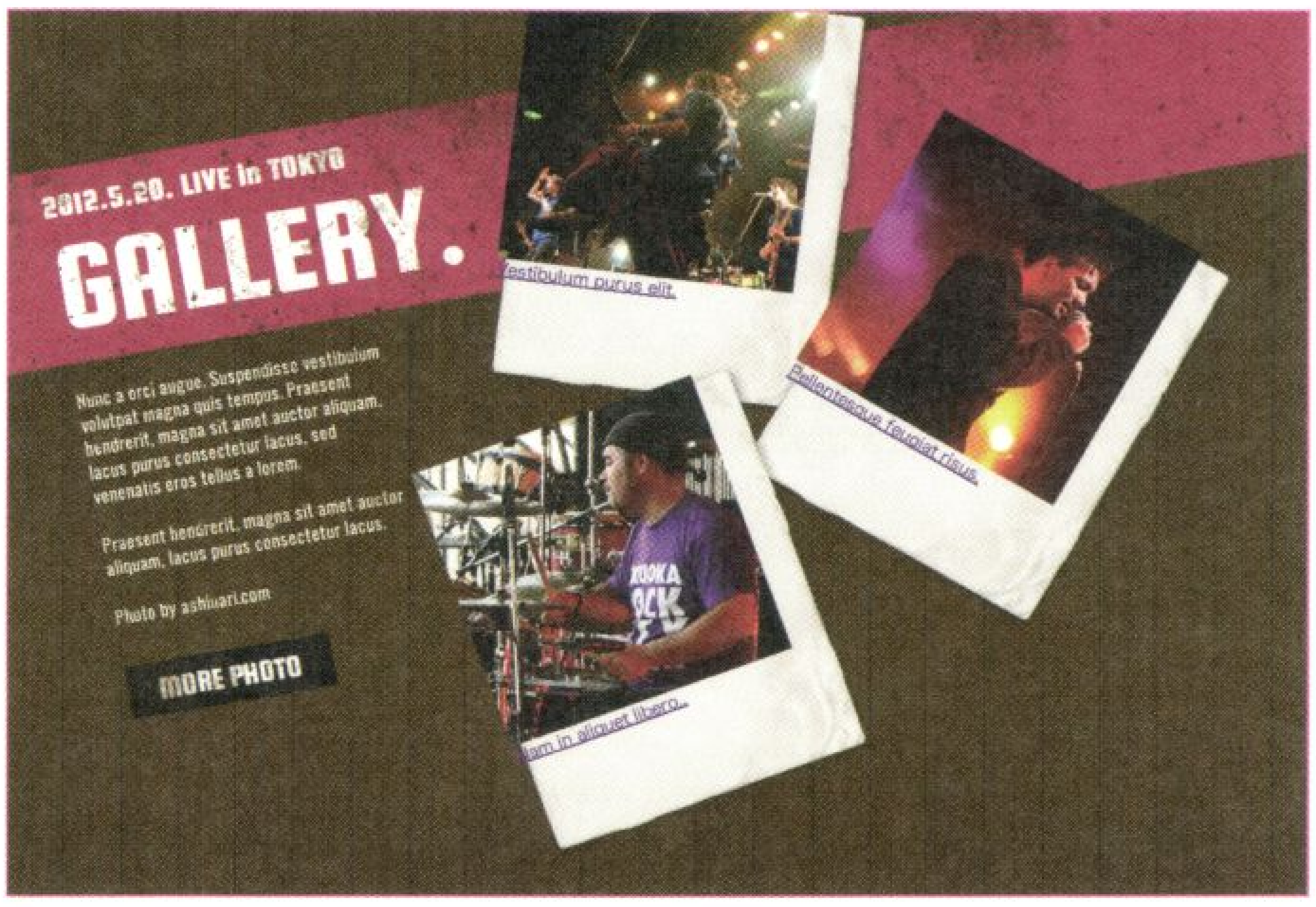
```
/*写真枠3点の共通スタイル*/
ul#thumb {
  list-style: none;
}
ul#thumb li {
  position: absolute;
  width: 328px;
  height: 404px;
  background: url(../images/photo_bg.png) no-repeat;
}

/*写真枠1*/
ul#thumb li.thumb1 {
  left: 470px;
  top: -20px;
  -webkit-transform: rotate(7deg) ;
— 省略 —
  transform: rotate(7deg) ;
}

/*写真枠2*/
ul#thumb li.thumb2 {
  left: 760px;
  top: 150px;
  -webkit-transform: rotate(30deg) ;
— 省略 —
  transform: rotate(30deg) ;
}

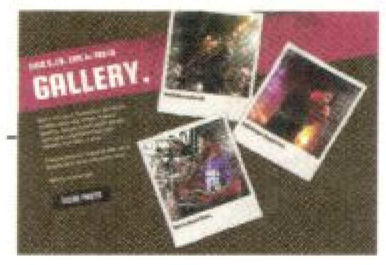
/*写真枠3*/
ul#thumb li.thumb3 {
  left: 440px;
  top: 380px;
  -webkit-transform: rotate(-20deg) ;
— 省略 —
  transform: rotate(-20deg) ;
}
```

08-2 写真ごとに角度と表示位置を変えている



ATTENTION!

IEはtransformに対応していないので回転させることはできませんが、コンテンツの可読性は確保されます。



09

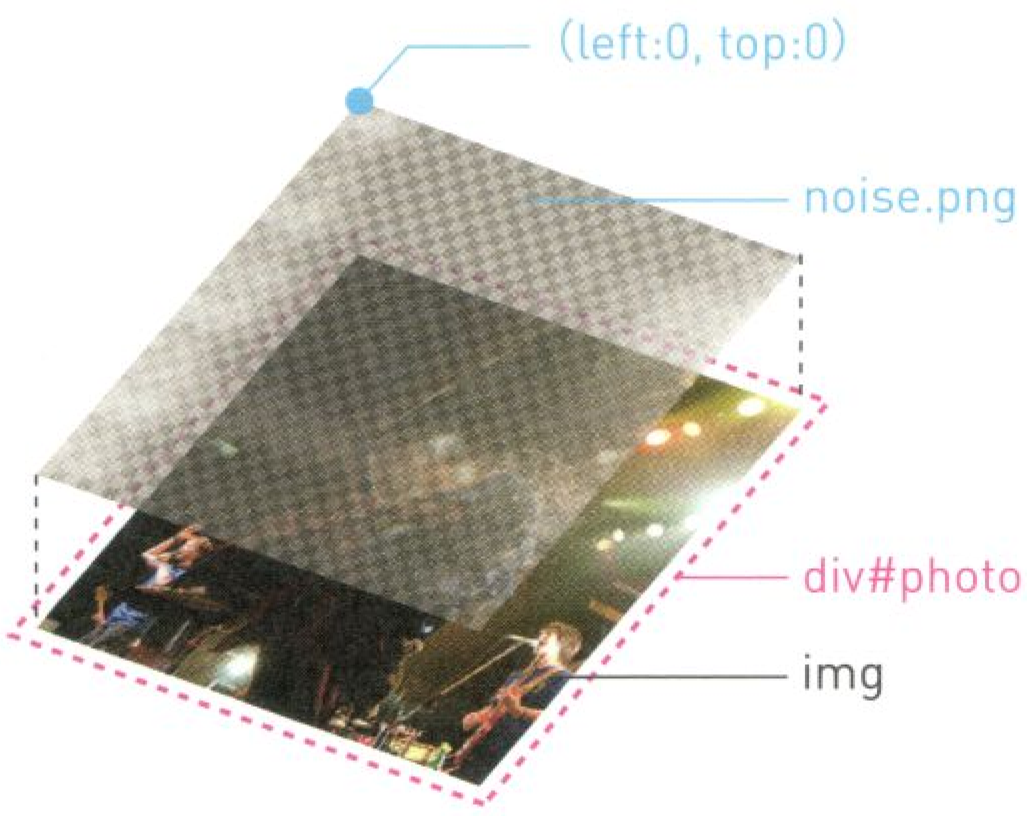
次に09-1を追記します。これは、写真枠の中に表示する写真のスタイルを設定します。まず共通スタイルとして、写真の位置や大きさを指定しています。ここで注目したいのがそのあとの処理です。「.photo」のdiv要素に擬似クラスのafterを使って掠れのテクスチャ画像をcontentとして追加し、それを絶対配置で写真の左上に揃えます09-2。afterで追加したcontentは内容よりも上に表示されるので、これによって写真の上に掠れのテクスチャを重ねることが可能です09-3 09-4。

09-1 写真のスタイル（style.css）

```
/*写真の共通スタイル*/
.photo {
  width: 280px;
  height: 280px;
  margin: 25px 0 0 22px;
  position: relative;
}

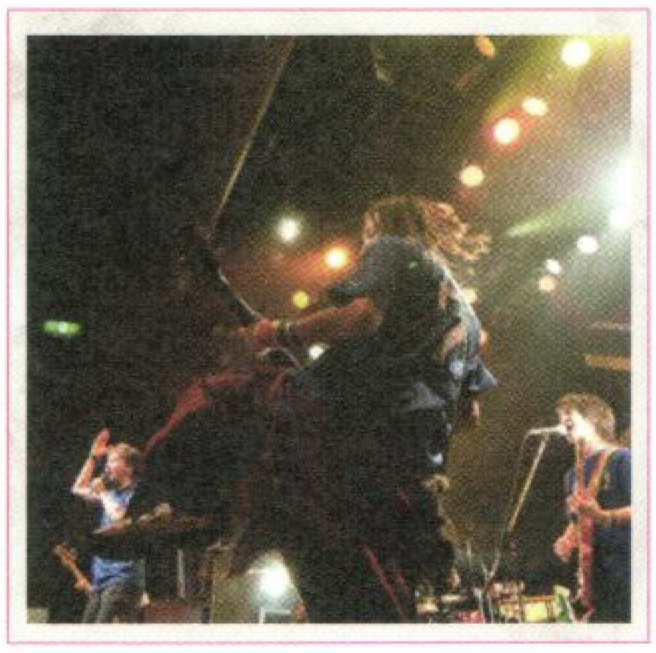
/*写真に掠れのテクスチャ画像を重ねる*/
.photo:after {
  content: url(../images/noise.png);
  position: absolute;
  width: 280px;
  height: 280px;
  left: 0;
  top: 0;
}
```

09-2 テクスチャを重ねる仕組み



noise.pngをdiv#photoに対して
擬似クラスの「:after」で挿入。
さらに、「position:absolute」で
div#photoの左上に揃える。

09-3 テクスチャを重ねる前



09-4 テクスチャを重ねたあと



> **ATTENTION!**
> IEはafterに対応していないので、掠れのテクスチャは後ほどJavaScriptで挿入します。

キャプションのスタイルの作成

10

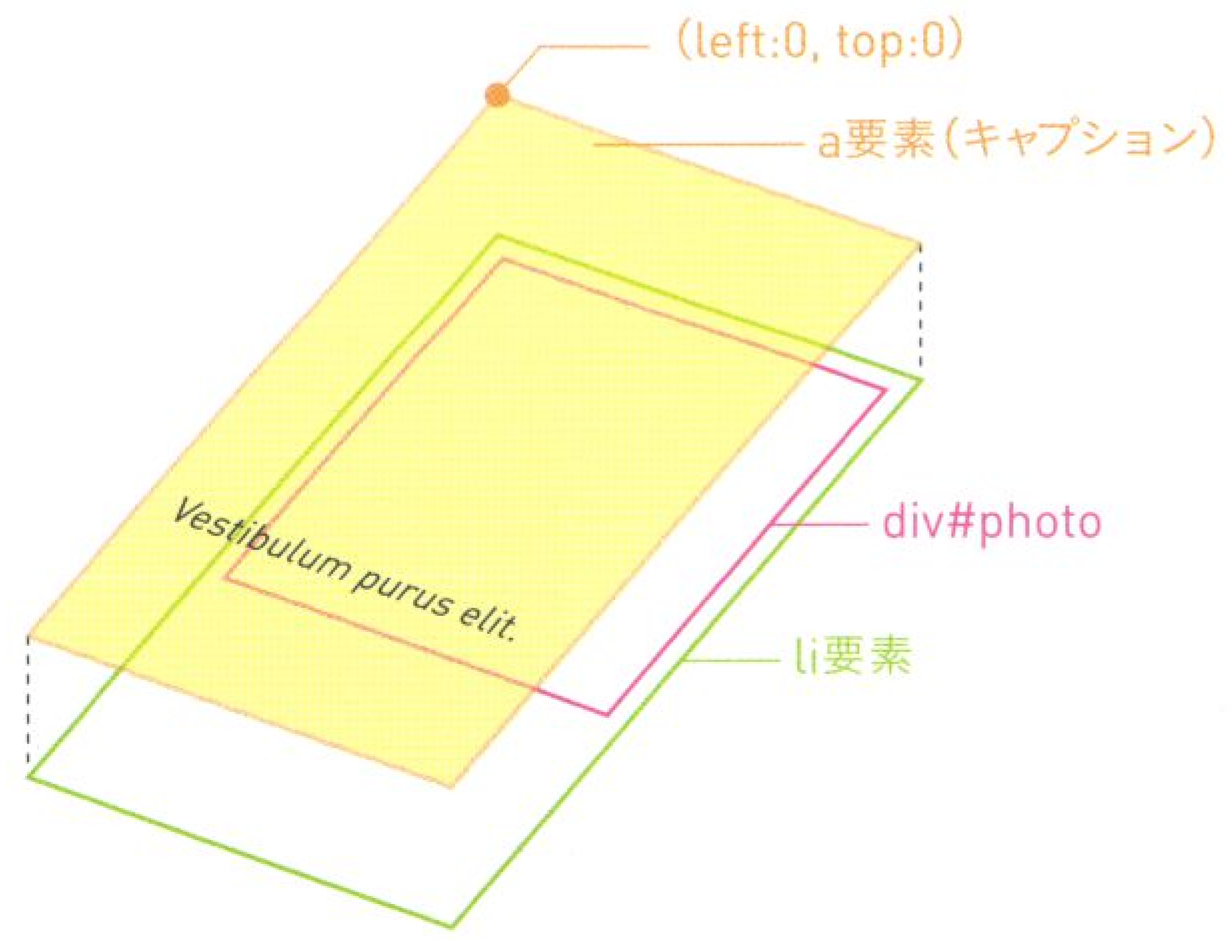
最後にキャプションのスタイル10-1を追記します。キャプションのテキストをa要素でマークアップしてありますが、これを10-2のようにli要素全体に広げることで写真枠全体をクリッカブルにしています。ここまでで基本のスタイルは完成です10-3。

10-1 キャプションのスタイル（style.css）

```
/*キャプションのリンクを写真枠全体に広げる*/
.caption a {
  display: block;
  position: absolute;
  left: 0;
  top: 0;
  padding: 314px 25px 0 22px;
  width: 281px;
  height: 90px;
  font-family: 'Oswald', sans-serif;
  text-decoration: none;
  color: #000;
  font-size: 15px;
}

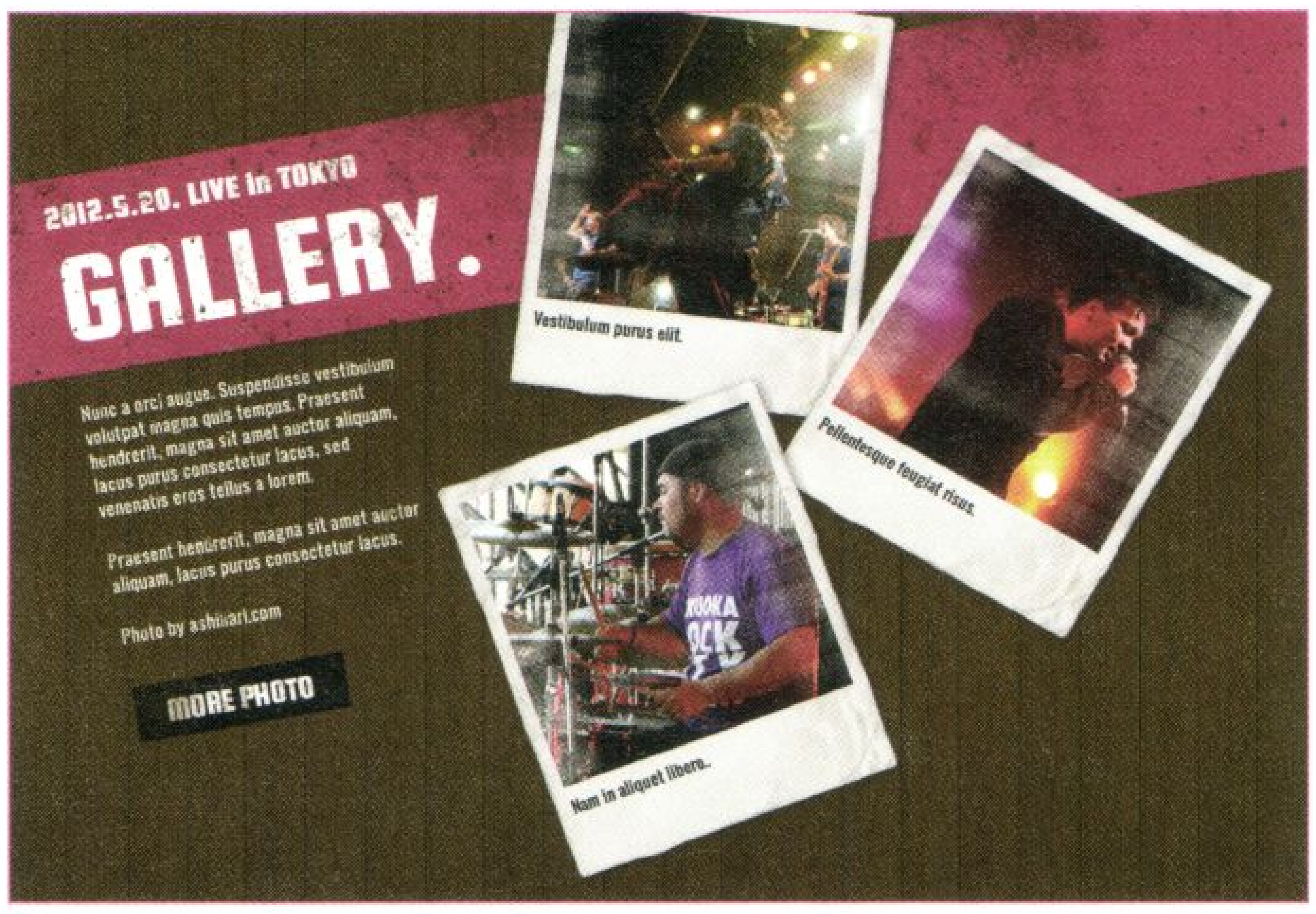
/*キャプションのhover*/
.caption a:hover {
  color: #f39;
}
```

10-2 写真枠全体をクリッカブルにする仕組み



キャプションのa要素を「display:block」にして
li要素を覆うように広げる。
さらに、「position:absolute」でli要素の左上に揃える。

10-3 完成イメージ



ATTENTION!

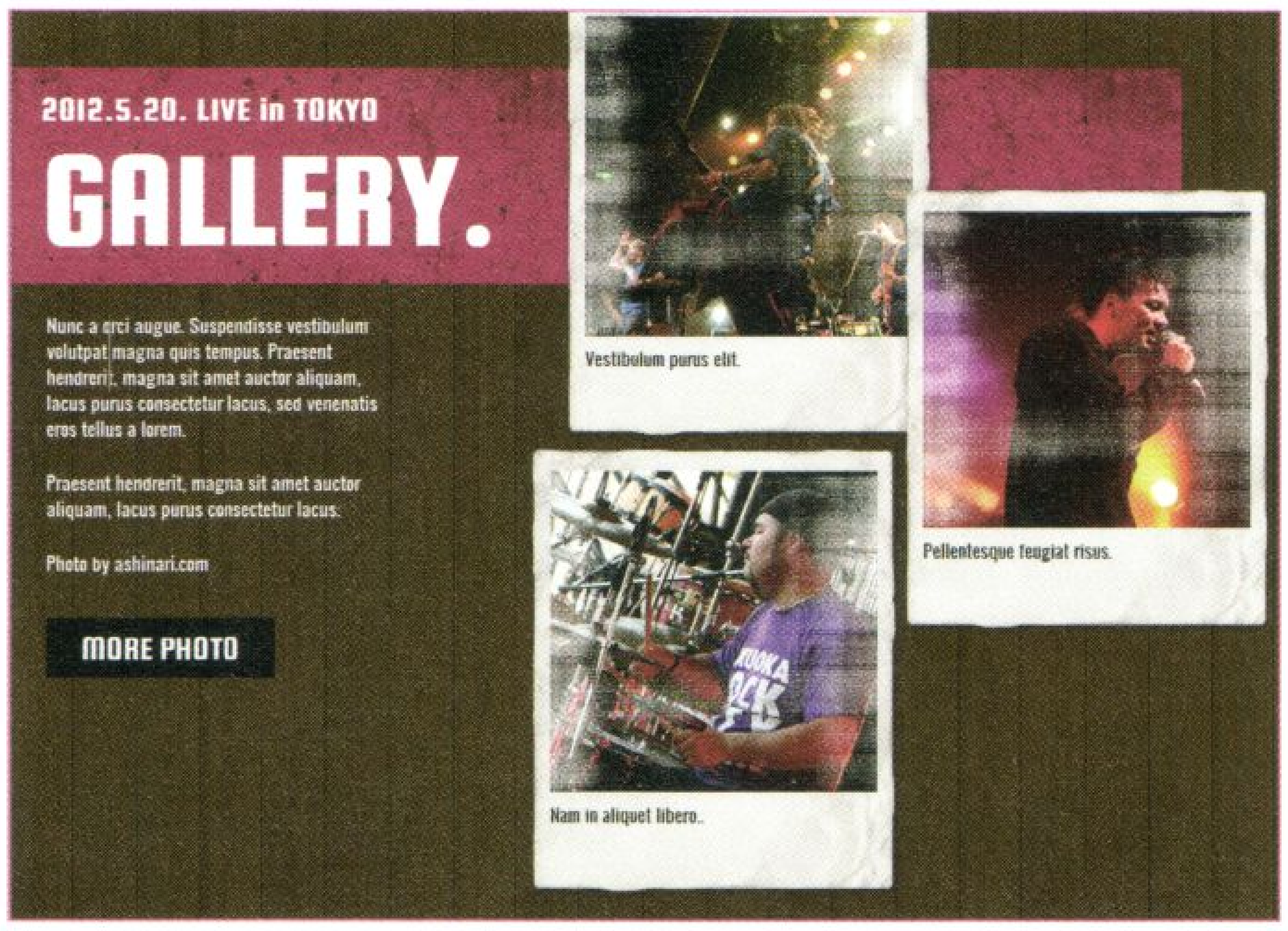
IEではa要素の下にテキストや画像があるとその部分はクリッカブルになりませんが、文字自体はクリックできるので、アクセシビリティに支障はありません。

IE対策用のスタイル、JavaScriptの作成

11

続いて、IE対策用のスタイルです。IE6～8で要素を回転させるのはさまざまな支障があり難しいため、今回はIE6～8のみ 11-1 のようにすべてを通常の角度で表示するレイアウトとします。このため、IE6～8のみに適用するスタイルを「styleforie.css」という別ファイルに記述してデザインを調節します。なお「styleforie.css」は、必ず「style.css」のあとに読み込むようにしましょう。

11-1 IE6～8用のレイアウト

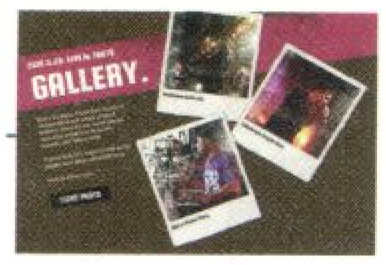


MEMO

IE6～8で要素を回転するには、cssSandpaper **01** などのJavaScriptプラグインを使う方法もありますが、今回の作例では一部に支障が出るため使用していません。

01 cssSandpaper

http://www.useragentman.com/blog/csssandpaper-a-css3-javascript-library/



12

「styleforie.css」に 12-1 を記入します。最初に、タイトルや写真を回転しなかったことによって生じる位置のずれを調整します。続いて、afterで挿入できなかった写真の掠れテクスチャをJavaScriptで挿入する際のスタイルを設定します。

12-1 styleforie.css

```
@charset "UTF-8";

/*位置のずれを調整*/
#header h1 {
  margin: 100px 0 0 -30px;
  background: transparent url(../images/
  mask.png);
}
#header p {
  padding: 25px 0 0 55px;
}

/*JavaScriptで追加する掠れテクスチャ画像
のスタイル*/
.photonoise {
  position: absolute;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  background: url(../images/noise.png)
  no-repeat;
}
```

13

HTMLへ戻り、head要素内で「styleforie.css」を読み込んでいるlink要素のあとに、 13-1 を追記します。
1行目はJavaScriptライブラリの「jQuery」 13-2 を読み込みます。
続く4行目で、after要素で挿入できなかった掠れテクスチャを写真のあとに追加します。
さらに、6行目で「DD_belatedPNG」を有効にするための処理を行います。この例では「transparent」というクラス名が付くpng画像の透過を有効にします。
7～9行目で、透過を有効にしたい要素に「transparent」のクラス名を追加しています。

13-1 HTMLのheadでJavaScriptを読み込む

```
01 <script type="text/javascript" src="js/jquery-1.7.1.js"></script>
02 <script type='text/javascript'>
03   $(document).ready(function() {
04     $('.photo img').after('<div class="photonoise"></div>');
05     if(navigator.userAgent.indexOf("MSIE 6")>=0){
06       DD_belatedPNG.fix('.transparent');
07       $('#header h1').addClass('transparent');
08       $('#thumb li').addClass('transparent');
09       $('.photonoise').addClass('transparent');
10     }
11   });
12 </script>
```

13-2 jQuery

http://jquery.com/

まとめ [CONCLUSION]

CSS3を使うことで、無駄な画像を作ることなく必要最小限の画像でアナログな質感を表現する手法が増えました。ブラウザによる差異も大きいのでまだまだ完全とはいえませんが、ポイントを絞って使うことで、効果的な表現が可能となります。

特に、afterを使った画像の多重表現は使いどころも多く、覚えておいて損のないテクニックです。ただし、アナログ表現もやりすぎは禁物。アクセントとして使うように心がけましょう。

08 テキストが読みやすい段組みレイアウト


芥川 竜之介

貝殻

一　猫

　彼等は田舎（ゐなか）に住んでゐるうちに、猫を一匹飼ふことにした。猫は尾の長い黒猫だつた。彼等はこの猫を飼ひ出してから、やつと鼠の災難だけは免（まぬか）れたことを喜んでゐた。

半年（はんとし）ばかりたつた後（のち）、彼等は東京へ移ることになつた。勿論猫も一しよだつた。しかし彼等は東京へ移ると、いつか猫が前のやうに鼠をとらないのに気づき出した。「どうしたんだらう？　肉や刺身（さしみ）を食はせるからかしら？［＃底本では「？」の後は１字アキ］」「この間Rさんがさう言つてゐましたよ。猫は塩の味を覚えると、だんだん鼠をとらないやうになるつて。」――彼等はそんなことを話し合つた末、試みに猫を餓ゑさせることにした。

しかし、猫はいつまで待つても、鼠をとつたことは一度もなかつた。そのくせ鼠は毎晩のやうに天井裏（てんじやううら）を走りまはつてゐた。彼等は、――殊に彼の妻は猫の横着（わうちやく）を憎み出した。が、それは横着ではなかつた。猫は目に見えて痩せて行きながら、掃（は）き溜（だ）めの魚（さかな）の骨などをあさつてゐた。「つまり都会的になつたんだよ。」――彼はこんなことを言つて笑つたりした。

　そのうちに彼等はもう一度田舎（ゐなか）住ひをすることになつた。けれども猫は不相変（あひかはらず）少しも鼠をとらなかつた。彼等はとうとう愛想（あいそ）をつかし、気の強い女中に言ひつけて猫を山の中へ捨てさせてしまつた。

すると或晩秋の朝、彼は雑木林（さふきばやし）の中を歩いてゐるうちに偶然この猫を発見した。猫は丁度（ちやうど）雀を食つてゐた。彼は腰をかがめるやうにし、何度も猫の名を呼んで見たりした。が、猫は鋭い目にぢつと彼を見つめたまま、寄りつかうとする気色（けしき）も見せなかつた。しかもパリパリ音を立てて雀の骨を噛み砕いてゐた。

二　河鹿

　或温泉にゐる母から息子（むすこ）へ人伝（ひとづ）てに届けたもの、――桜の実（み）、笹餅、土瓶（どびん）へ入れた河鹿（かじか）が十六匹、それから土瓶の蔓に結（むす）びつけた走り書きの手紙が一本。
　その手紙の一節はかうである。――「この河鹿（かじか）は皆雄（をす）に候。雌（めす）はあとより届け候。尤（もつと）も雌雄（めすをす）とも一つ籠に入れぬやうに。雌は皆雄を食ひ殺し候。」

誕生	1892年3月1日 日本・東京市京橋区入船町8丁目
死没	1927年7月24日（満35歳没）
職業	小説家
言語	日本語
国籍	日本
教育	学士（文学）
最終学歴	東京帝国大学英文科
ジャンル	短編小説
主題	近代知識人の苦悩
文学活動	新現実主義
代表作	『羅生門』（1915年） 『鼻』（1916年） 『戯作三昧』（1917年） 『地獄変』（1918年） 『藪の中』（1922年） 『河童』（1927年） 『歯車』（1927年）
配偶者	塚本文（1919年 - 1927年）
子供	芥川比呂志（長男） 芥川多加志（次男） 芥川也寸志（三男）
親族	塚本善五郎（義父） 芥川貴之志（孫） 芥川麻実子（孫）

「芥川 竜之介」（2012年1月8日（日）UTC の版）『ウィキペディア日本語版』。

テキスト主体のWebページを制作する際、大事なのは文章自体のレイアウトに加えて、見出しや本文の強弱に気をつけることです。column-countプロパティを使うことで段組みを表現でき、より読みやすくすることができます。

制作・文／藤本勝己（株式会社レリーフ）

制作のポイント

- line-heightで行間を設定
- 見出しと本文に強弱をつける
- column-countを使って段組みを表現する

対応ブラウザ

IE	P.E.
Safari	5.1 以上
Firefox	9 以上
Chrome	16 以上

TECHNIC 1

強調したい場所を見極める

欲張って多くの文章を目立たせようとすると、かえって強調したい場所がわかりにくくなります。優先すべき箇所を見極めて装飾しましょう。



TECHNIC 2

行間や余白のバランスに配慮する

写真やイラスト等を追加して、文字だけにならないよう工夫しましょう。行間や余白を取ることも重要です。





見出しや読ませたい部分を目立たせる

01

テキストベースのページ 01-1 を装飾して、見栄え良くしていく内容を説明します。

01-1 テキストだけで構成されたページ



02

見出しにする部分に、フォントサイズ、フォント色、背景・囲い等の装飾を使い分けて目立たせていきます 02-1 02-2。

02-1 装飾を施したページ



MEMO

02-2 の HTML はコーディングの途中段階のもののため、完成後の HTML のソースコードでは一部変更されています。

02-2 HTML の記述

```
<body>
<h1>芥川 竜之介</h1>
<table>
  <tr>
    <th scope="row">誕生</th>
    <td>1892年3月1日<br />
      <a href="http://ja.wikipedia.org/wiki/
      %E3%83%95%E3%82%A1%E3%82%A
      4%E3%83%AB:Flag_of_Japan.svg"
      title="日本の旗"></a> 日本・東京市京
      橋区入船町8丁目</td>
  </tr>
  <tr>
    <th scope="row">死没</th>
    <td>1927年7月24日（満35歳没）<br />
    </td>
  </tr>
  <tr>
    <th scope="row">国籍</th>
    <td>日本</td>
  </tr>
  <tr>
    <th scope="row">教育</th>
    <td>学士（文学）</td>
  </tr>
  <tr>
    <th scope="row">最終学歴</th>
    <td>東京帝国大学英文科</td>
  </tr>
  <tr>
    <th scope="row">ジャンル</th>
    <td>短編小説</td>
  </tr>
  <tr>
    <th scope="row">主題</th>
    <td>近代知識人の苦悩</td>
  </tr>
</table>
「<a href="http://ja.wikipedia.org/wiki/芥川
龍之介" target="_blank">芥川 竜之介</a>」
（2012年1月8日 (日) UTCの版）『ウィキペデ
ィア日本語版』。
<hr />

<h2>貝殻</h2>
<h4>一　猫</h4>
── 省略 ──
```

＊COLUMN

目立たせるコンテンツを絞り込む

特に読んでほしい部分がある場合は、優先度が低いコンテンツを控えめに作り、本当に読ませたい部分を目立たせるようにバランスをとります。
01 の例のように赤文字を多用しすぎると、見づらくなってしまいますので、最小限にするのがベストです。

01 赤文字が多すぎてわかりにくい例（左）と、赤文字を最小限にとどめた例（右）





h2要素をtext-shadowで装飾

03

h2要素（ここでは作品名）にtext-shadowプロパティで装飾して強調します。text-shadowは、テキストに対して影をつけることができるプロパティです。
値には、shadowを指定します。shadowは「,」カンマで区切ることができ、複数指定も可能です 03-1。

「,」カンマ区切りで複数の値が列挙された場合、先に指定された値が上に、後に指定された値が下に重なって表示されます。

text-shadowプロパティにより表示された影の部分は、レイアウトに影響を及ぼさず、スクロールバーも発生させません。

03-1 text-shadowプロパティの基本的な記述

```
text-shadow : 2px 4px 6px #666666;
```

横方向のずれ（2px）
縦方向のずれ（4px）
ぼかし具合（6px）
影の色（#666666）

04

影の指定はtext-shadowプロパティの場合、2つ〜4つの長さと色が一組となります。例えば「2px 2px 3px #999」で一組の指定になります 04-1 04-2 04-3。
（box-shadowプロパティでの指定とは一部異なります。）
ただし、ブラウザによっては4つ目の長さ（影の広がり幅）に対応していないなど、完全に対応している環境ばかりではないため、基本的には必ず長さを3つ指定しておくといいでしょう。

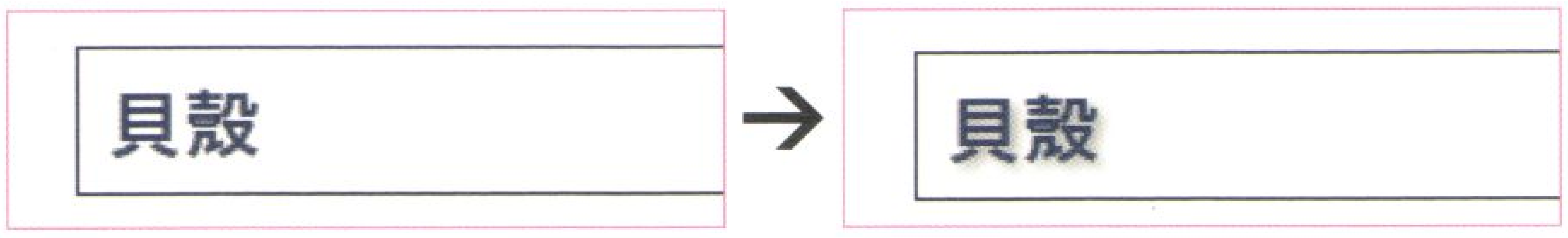
04-1 CSSの中でのtext-shadowの記述例

```
h2 {
  padding: 8px;
  border: 1px solid #000066;
  color: #003366;
  text-shadow:2px 2px 3px #999;
}
```

04-2 HTML

```
<h2>貝殻</h2>
```

04-3 文字に影がついた



カラムを設定して区切り線を表示

05

column-countを使うと、長い文章を段組みで表示することができます。
column-countプロパティの値には、1以上の整数を指定します。それにより、カラムの数が決定されます 05-1。
column-widthプロパティが指定されていない場合には、カラム幅も自動で決定されます。

05-1 column-countプロパティの基本的な記述

```
column-count : 2;
```

カラムの数（2）

MEMO

今までは少々手間の掛かった横並びの段組み変更がcolumn-countを利用することによって、数字（値）を変えるだけで簡単に行えるようになります。ただし、あまり分割しすぎると逆に読みにくくなることがあるので、段組みを増やしすぎないよう注意してください。また「column-count:auto」とした場合には、ほかのプロパティにより自動で段落数が決定されます。



06

Firefox、Chrome、Safariで表示させるために、-moz-、-webkit-を付け足したものも、同じように記述します06-1 06-2。表示すると06-3のようになります。

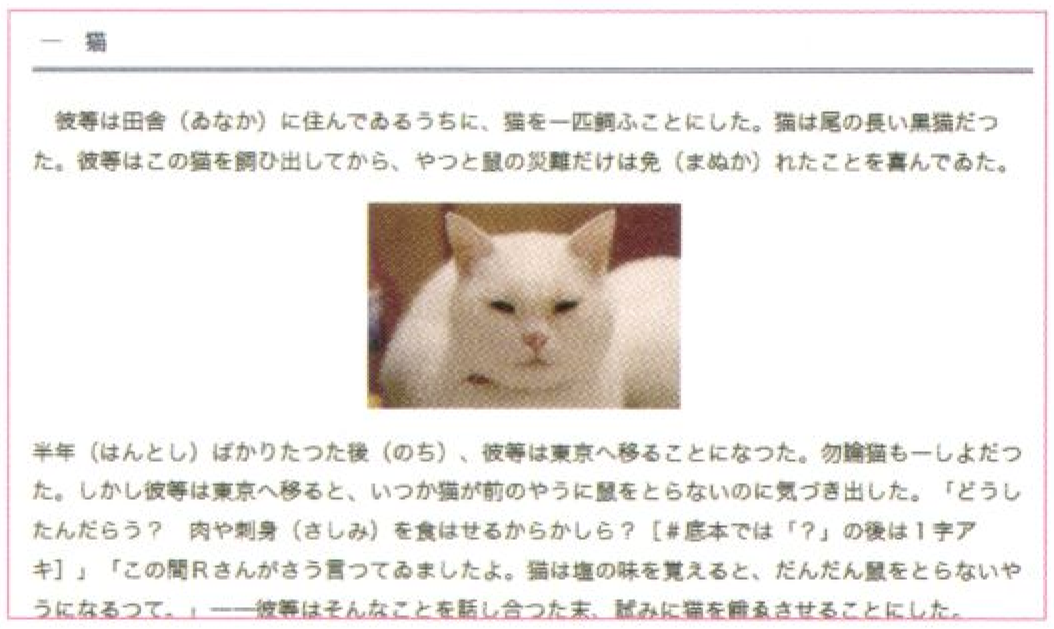
06-1 CSSでの段組指定

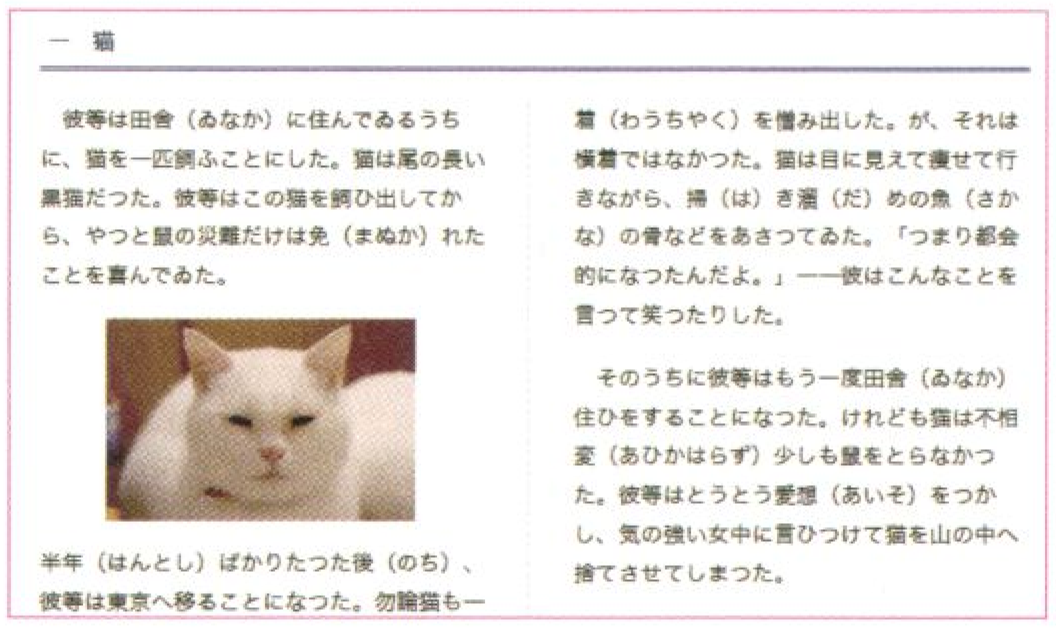
```
.dankumi{
  -webkit-column-count:2;
  -moz-column-count:2;
  column-count:2;
— 省略 —
}
```

06-2 HTML

```
<div class="dankumi">
  <p>　彼等は田舎（ゐなか）に住んでゐるうちに、猫を一匹飼ふことにした。猫は尾の長い黒猫だつた。彼等はこの猫を飼ひ出してから、やつと鼠の災難だけは免（まぬか）れたことを喜んでゐた。</p>
  <p class="photo"><img src="images/photo01.jpg" alt="猫の写真" width="200" height="133" /></p>
— 省略 —
</div>
```

06-3 1段組でこの行の長さだと読みにくいが（左）、2段組にすると読みやすくなる（右）





07

次に、column-ruleプロパティでカラムとカラムの間に区切り線を設定します。
column-ruleプロパティを利用すると、column-rule-width（線幅）、column-rule-style（線種）、column-rule-color（色）を一括で指定することができます。column-ruleプロパティの値には、半角でスペースで区切って各プロパティの値を指定します。指定方法は07-1のようになります。

07-1 column-ruleプロパティの基本的な記述



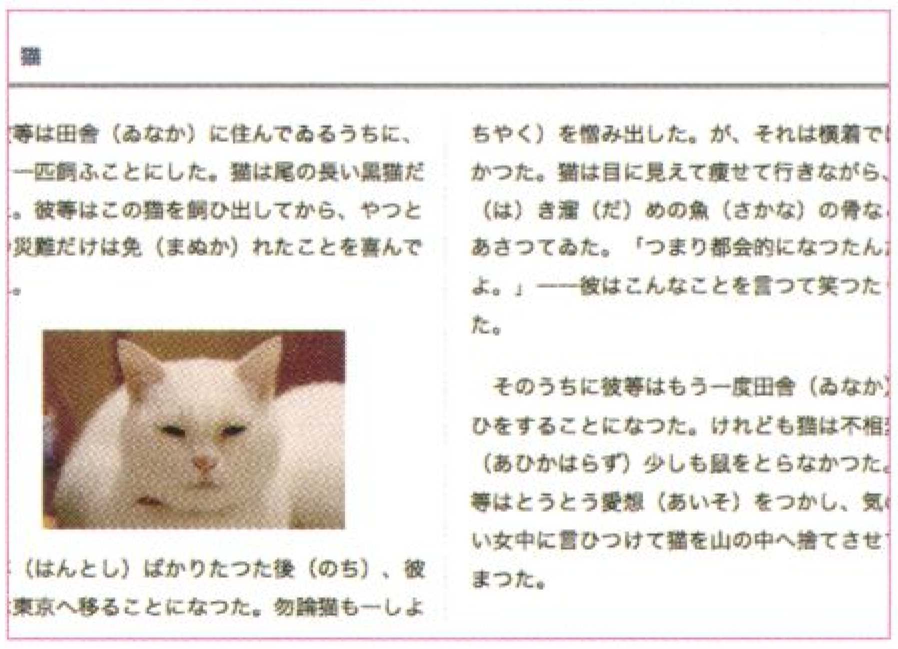
08

ここではFirefox、Chrome、Safariで表示させるために、-moz-、-webkit-を付け足したものも同じように記述します08-1。08-2のようにカラムの間に区切り線が加わり、長い文章の可読性が上がります。

08-1 CSSでの区切り線の指定

```
.dankumi{
— 省略 —
-webkit-column-rule: 1px dotted #ccc;
-moz-column-rule:1px dotted #ccc;
column-rule:1px dotted #ccc;
— 省略 —
}
```

08-2 カラム間に区切り線が引かれた



column-gapでカラム間隔を調整する

09

column-gapプロパティを使って、カラム間の幅を決定することができます。column-gapプロパティの値には、間隔を数値で指定するか、または「normal」を指定します09-1。

09-1 column-gapプロパティの基本的な記述

```
column-gap : 40px;
```

（40px：カラム間の幅）

10

ここでは10-1のように指定しました。Firefox、Chrome、Safariで表示させるために、-moz-、-webkit-を付け足したものも同じように書きます。これで10-2のようにカラムの間隔が広がります。

10-1 CSSでのカラム間の距離の指定

```
.dankumi{
— 省略 —
-webkit-column-gap:40px;
-moz-column-gap:40px;
column-gap:40px;
}
```

10-2 カラム間の距離が指定された



*COLUMN

Internet Explorerへの対応方針

column-countプロパティはInternet Explorerには対応していないため、段組みの指定が適用されませんが、コンテンツが読めなくなるわけではありません。
もしCSSハックを使うことに問題がなければ、IEだけ左右の余白を取るなどして、横幅が長くなりすぎないようにします01。

01 IE向けに左右の余白を取り、横幅を抑える



まとめ [CONCLUSION]

column-countプロパティは段組みの変更や調整がしやすく、使いやすい仕様になっています。
長い文章を快適に読んでもらうためには、行間・余白・行の長さなどに気をつける必要があります。強調したい部分が多くなりすぎないよう優先順位をつけて、全体のデザインにメリハリを出すことが重要です。

CHAPTER 02

ユーザー インターフェイス

User Interface

CSSだけで実装する横並びのプルダウンメニュー
表示内容を切り替えるタブナビゲーションの実装
画像を使わないで作るパンくず型の購入フロー
ユーザビリティに配慮した問い合わせフォーム
ul要素でシンプルに作る縦並びのプルダウンメニュー
ロールオーバーによるナビゲーションの演出
ユーザビリティの高いiPhone風UIフォーム
ページをスライドさせるリスト型ナビゲーション
カスタムURLスキームと文字量の変化に対応できるリスト

01 CSSだけで実装する横並びのプルダウンメニュー



JavaScriptを使わなくても、CSSだけで作成できるプルダウンメニューを紹介します。
WebKit系のみの対応となりますが、するっと動くモーションも付けることができ、「軽いのにリッチな表現」が可能です。

制作・文／佐藤とも子・サチコ（螺旋デザイン）

制作のポイント

- ul要素とli要素を使ったマークアップ
- :hoverの応用でメニューを作成
- transitionでプルダウンの動きを表現

対応ブラウザ

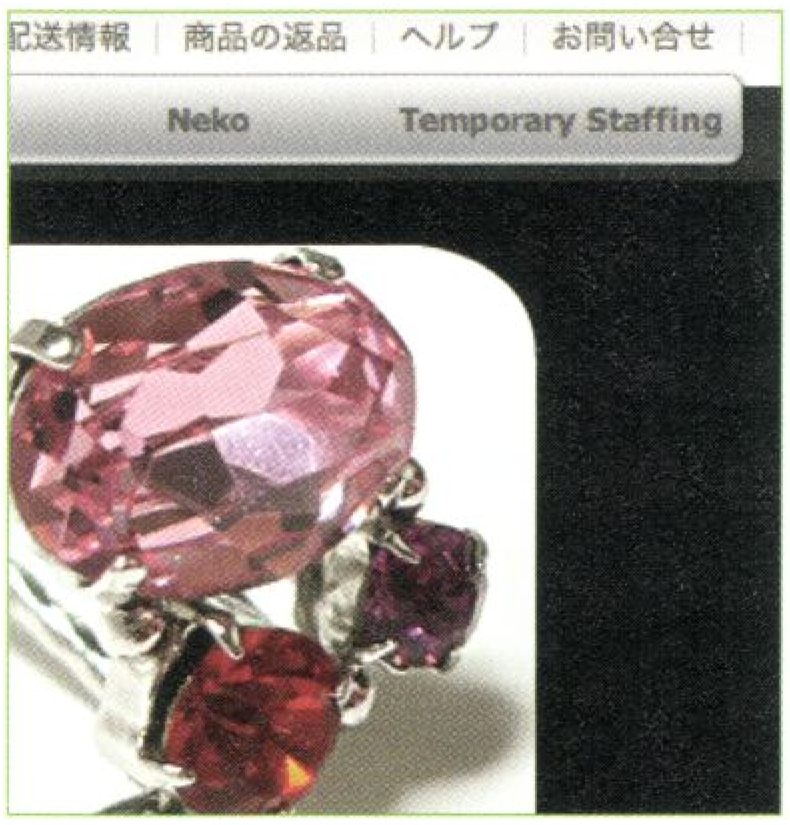
IE	8以上
Safari	5以上
Firefox	8以上
Chrome	16以上

※IE、Firefoxではtransitionに非対応。

TECHNIC 1

ボーダーと角丸の表現

:first-childと:last-childを利用して、メニュー部分の上下左右にボーダーをつけ、左端と右端だけ角丸の形状にしています。



TECHNIC 2

子メニューをコンテンツの前面に表示

positionとz-indexを利用して、親メニューの下・コンテンツの上に表示されるように調整します。





メニューのHTMLを作成

01

メニューのHTMLは、親のliの中に子メニューのulが入っている状態です 01-1。
CSSが適用されない表示でも、親子状態がわかるようになっています 01-2。

01-1 メニューの基礎となるHTML

```
<ul id="itemMenu">
<li><a href="#">Power Stone</a>
<ul class="sub">
<li><a href="#">Lapis Lazuli</a></li>
<li><a href="#">The Philosopher's Stone</a></li>
<li><a href="#">Crystal</a></li>
<li><a href="#">Cursed Gem</a></li>
<li><a href="#">Flying Stones</a></li>
</ul></li>
<li><a href="#">Cursed Items</a>
<ul class="sub">
<li><a href="#">Cleveland</a></li>
<li><a href="#">Charm</a></li>
<li><a href="#">Straw Man Set</a></li>
<li><a href="#">Cursed Gem</a></li>
</ul></li>
<li><a href="#">Books</a></li>
<li><a href="#">Magic Square</a></li>
<li><a href="#">Neko</a></li>
<li><a href="#">Temporary Staffing</a></li>
</ul>
```

01-2 CSSを切っても親子関係がわかる

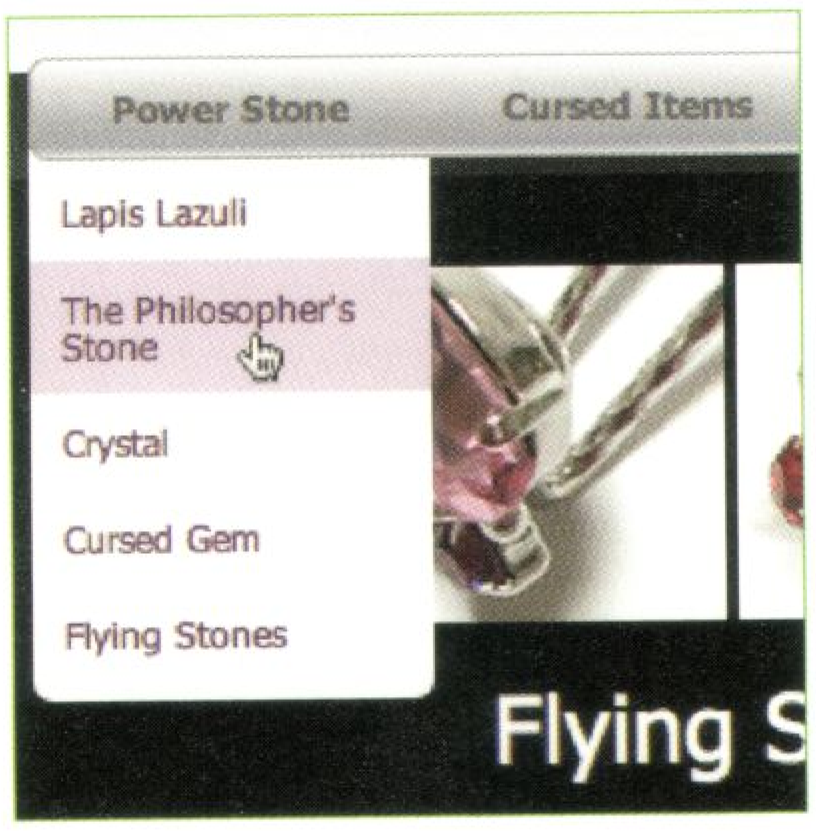


メニューのデザインをCSSで作る

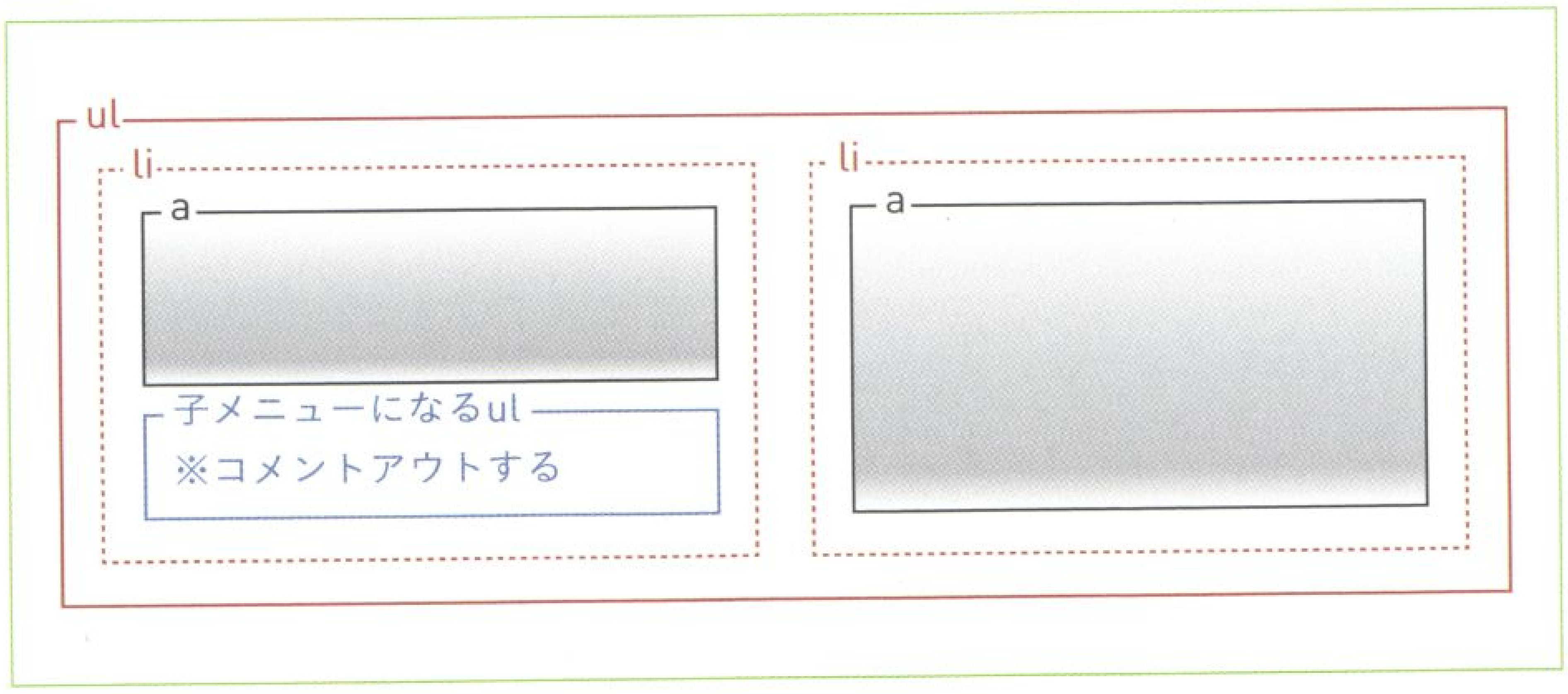
02

まずはプルダウンの見た目だけを先に作りましょう 02-1。子メニューとして降りてくる要素は今のところ邪魔なので、コメントアウトしておきましょう。親メニューの構成は、02-2のようになっています。ポイントは、ulにborderなどロールオーバー時に色を変えたい部分は設定しないことです。
また、レイアウト上はulを必ずしもfloatさせる必要はありませんが、ここでは子要素のliにしっかりとフィットするようにfloat:right;させています 02-3。このサイトのデザインでは右に揃っている要素が多いので、数ピクセルのずれが出た場合、右にfloatさせることで誤差がわかりにくいようにしています 02-4。

02-1 プルダウンメニューの見た目



02-2 親メニューの構成



02-3 ulを右寄せにする

```
#mainContent ul#itemMenu {
font-weight:bold;
line-height: 1;
float:right;
font-family: Tahoma, Geneva, sans-serif;
margin-bottom: 30px;
border-radius:6px;
/*textの影------------------------*/
text-shadow:0px 0px 2px rgba(0,0,0,0.45);

/*boxの影------------------------*/
box-shadow:1px 1px 3px rgba(0,0,0,0.4);
}
```

02-4 左右のレイアウトを比較

左側はメニューと縦のラインを感じさせるオブジェクトはありません。

右側はログインボタンなどグリッドを感じるデザインになっています。



03

liには、float:leftとメニューの横幅の設定だけしています 03-1。その子要素のaにスタイリングをしていきます 03-2。

グラデーションとボーダーをロールオーバーで切り替える設定をしていきます。

03-1 liの初期状態

```
#mainContent ul#itemMenu li {
float:left;
width: 130px;
}
```

03-2 liの子要素のaにスタイリングを加える

```
#mainContent ul#itemMenu li a {
font-size: 12px;
color: #666;
text-decoration: none;
text-align:center;
padding: 0.9em 0;
display:block;

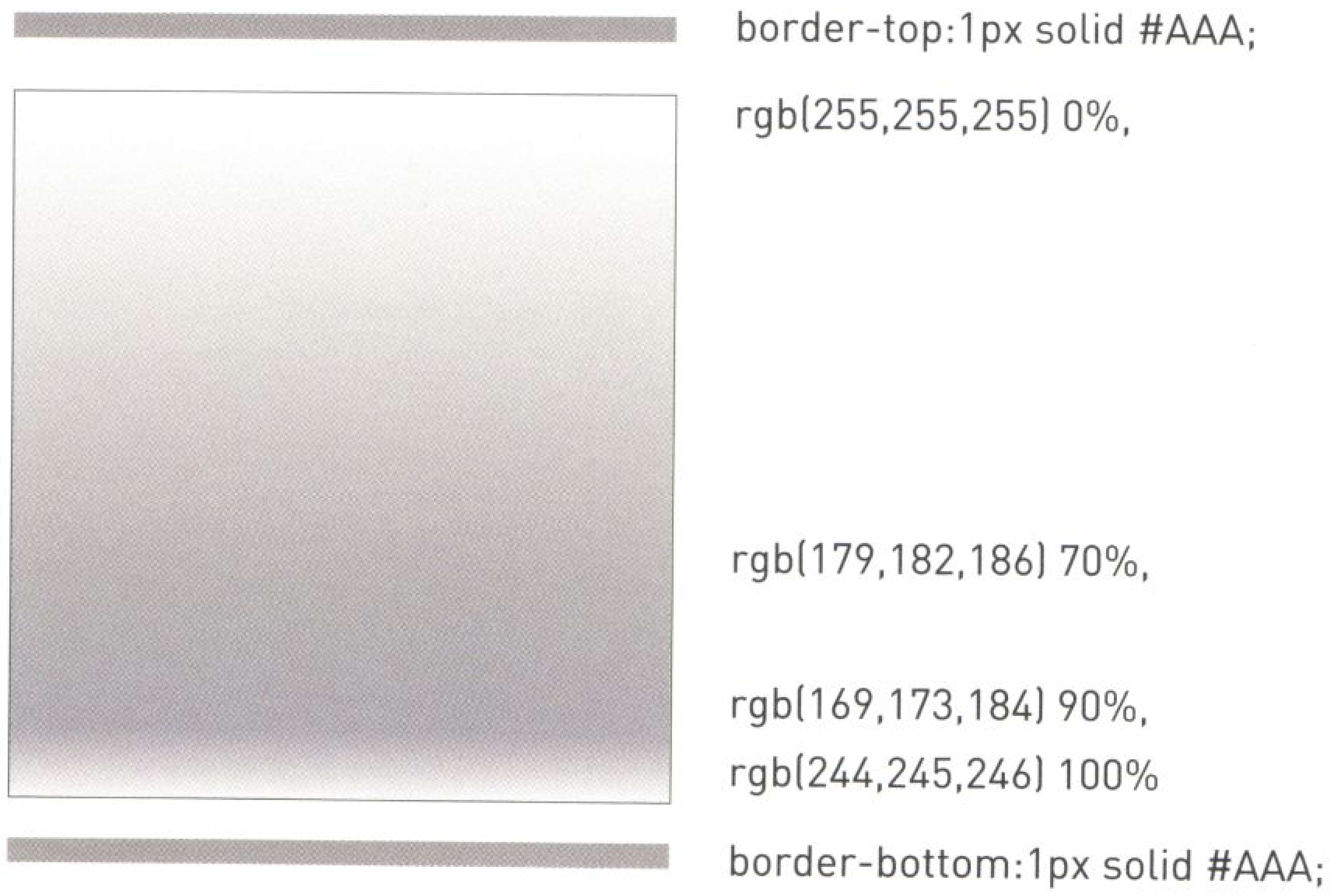
border-top:1px solid #AAA;
border-bottom:1px solid #AAA;
— 省略 —
```

04

引き続き、aタグにボタンのスタイリングをしていきます。display:blockにして、グラデーションの指定と上下のボーダーを指定します 04-1。

これがメニューボタンのアピアランスの基本になります。

04-1 グラデーションとボーダーの指定





05

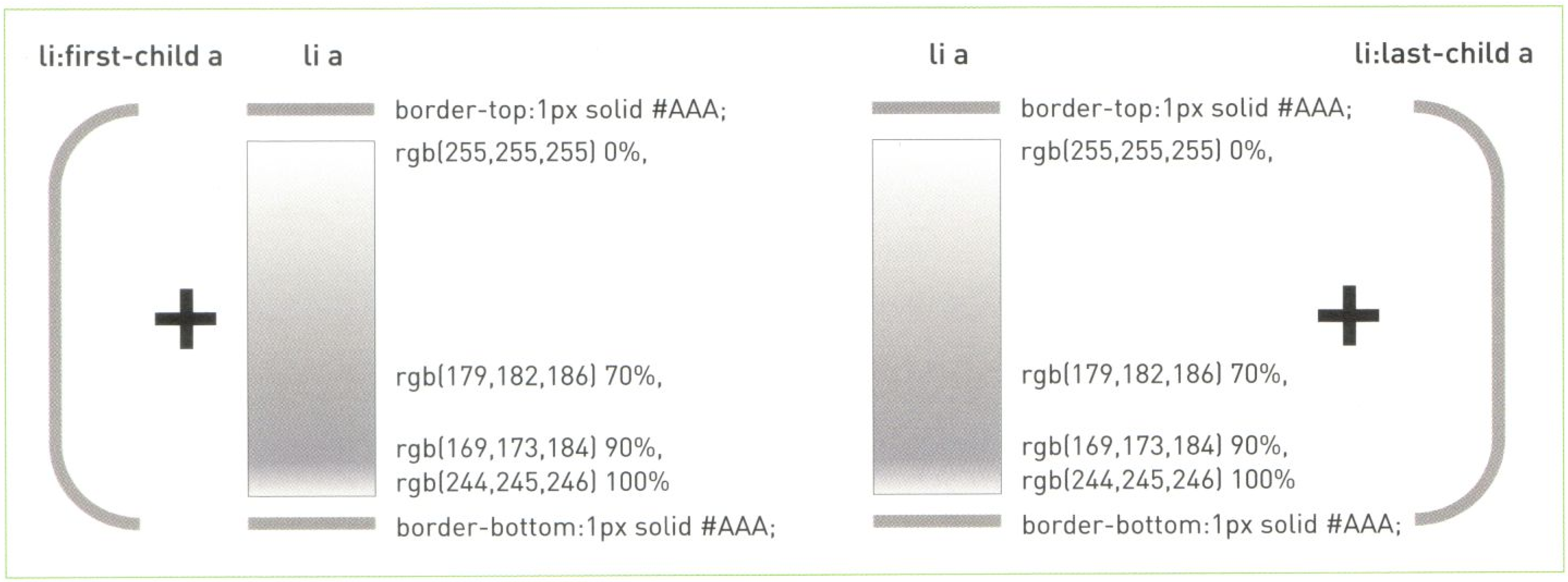
メニューの左端と右端を角丸の形状にして、ボーダーを両端に付けたいので 05-1、:first-child と :last-child を利用して設定しておきます 05-2。

05-2 :first-child と :last-child で両端にボーダーを付ける

```
#mainContent ul#itemMenu li:first-child a {
border-left:1px solid #AAA;
border-radius:6px 0 0 6px;
}
```

```
#mainContent ul#itemMenu li:last-child a {
border-right:1px solid #AAA;
border-radius: 0 6px 6px 0;
}
```

05-1 角丸とボーダーの設定



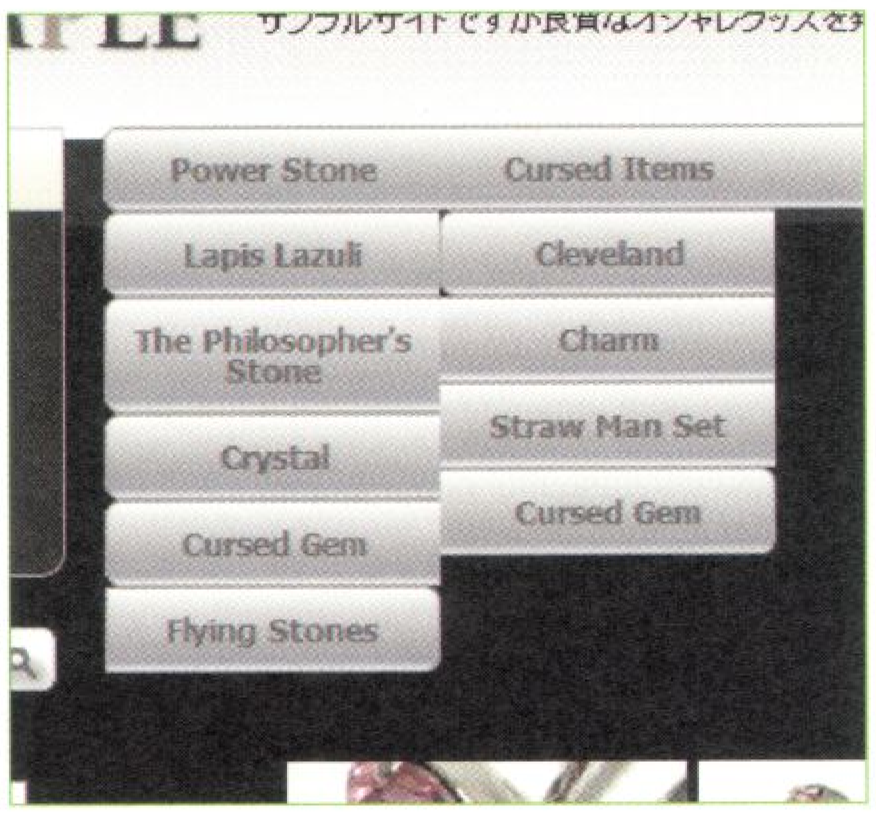
06

プルダウンで降りてくる、子メニューのCSSを作ります 06-1。
02 で付けていた、HTMLのコメントアウトを削除しましょう。
すると、親メニュー用に作ったa要素のスタイリングが影響しているはずです 06-2。これらを上書きして変更しましょう 06-3。

06-1 子メニュー



06-2 親メニューのスタイルが子メニューに影響している



06-3 親メニューによるスタイリングの影響を上書き

```
#mainContent ul#itemMenu li ul.sub li {
float:none;
}
#mainContent ul#itemMenu li ul.sub li a {
font-size: 12px;
color: #752A5A;
text-align:left;
padding: 0.9em 10px;
font-weight: normal;
background-image:none;
background-color: #FFF;
text-shadow:none;
border: none;
border-radius: 0;
}
```

07

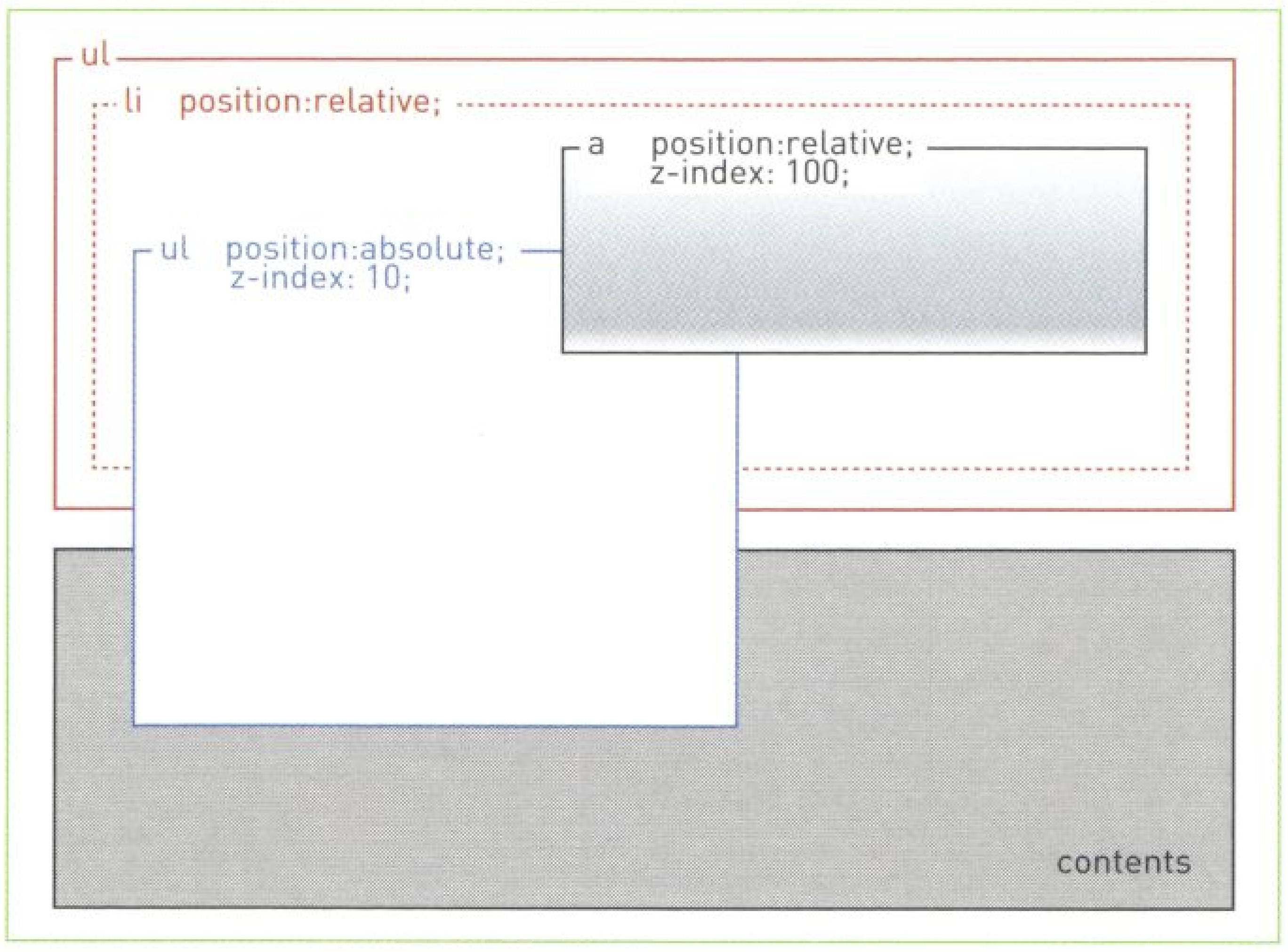
さらに、子メニュー用のデザインをします。降りてくるメニューがコンテンツ（商品紹介）の下に表示されないよう、positionとz-indexを使用して調整します 07-1。

ul.sub（子メニュー）だけposition:absoluteで指定しているのは、ページのコンテンツに影響が出ないように、浮いた状態にしたいからです。

MEMO

position:absolute（絶対位置）での位置指定は、基準点がどこになっているのか意識しておくことが大事です。
特に下準備をしていない場合はhtml要素を基準にします。この場合、今回のデザインのようにページを中央に配置するとウィンドウの横サイズの変動にメニューがついてきません。
position:absoluteにしたい要素の親要素にposition:relative（相対位置）を指定しておくと親要素を基準に位置が指定できるようになり、親要素の位置が別の要素の影響を受けて移動しても、それに連動するので非常に便利です。

07-1 メニューとコンテンツの位置関係



07-2 子メニューの位置を設定する

```
#mainContent ul#itemMenu li {
— 省略 —
position:relative;
}
#mainContent ul#itemMenu li a {
— 省略 —
z-index: 100;
position:relative;
}
```

```
— 省略 —

#mainContent ul#itemMenu li ul.sub {
— 省略 —
position:absolute;
z-index: 10;
— 省略 —
}
```

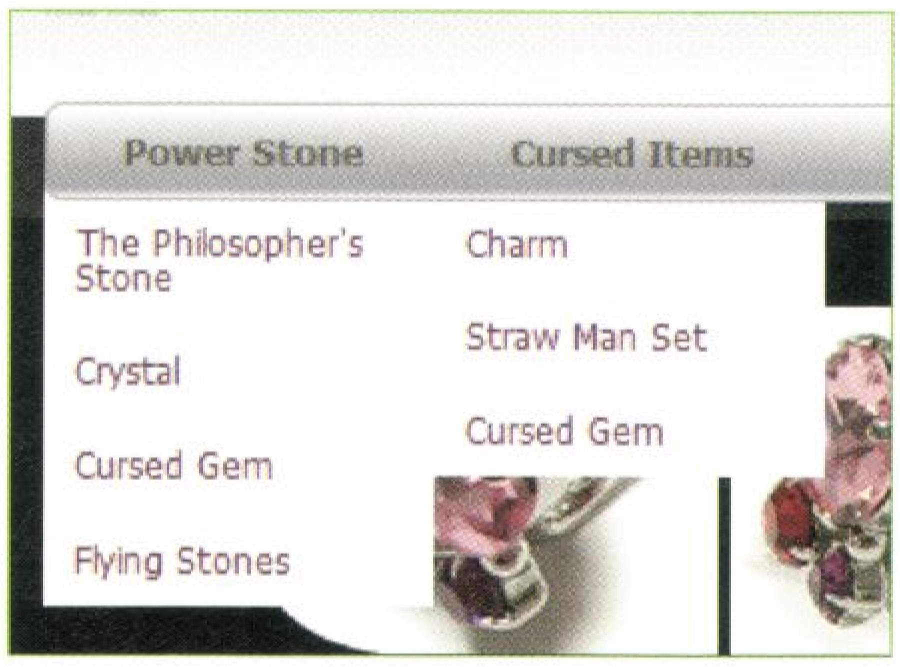
08

メニューの下から子メニューが表示されたとき、子メニューの一部が親メニューの背面に食い込み、見えなくなっています 08-1。これは、子メニューをposition:absoluteで設定しているからです。top方向に、リストの高さ分の値を入れましょう。

食い込みはなくなりましたが、今度ははじめのメニューの角丸部分に隙間ができてしまっています 08-2。

これは、topの値＋padding-topで対応しましょう 08-3。

08-1 子メニューが親メニューに食い込んでいる



08-2 メニューの角丸部分に隙間ができている



08-3 メニューの隙間をなくす

```
mainContent ul#itemMenu li ul.sub {
— 省略 —
top: 30px;
padding-top: 5px;
— 省略 —
}
```



09

メニューの一番下の角丸も、子メニューのul要素で設定しておきます 09-1。あとは、ロールオーバーの色合いなどを調整します 09-2。
これで見た目はOKです。

09-1 メニューの一番下の角丸を設定

```
#mainContent ul#itemMenu li ul.sub {
position:absolute;
z-index: 10;
font-size: 10px;
top: 30px;
padding-top: 5px;
padding-bottom: 6px;
background: #FFF;
border-radius:0 0 6px 6px;
}
```

09-2 ロールオーバーの色合いを調整

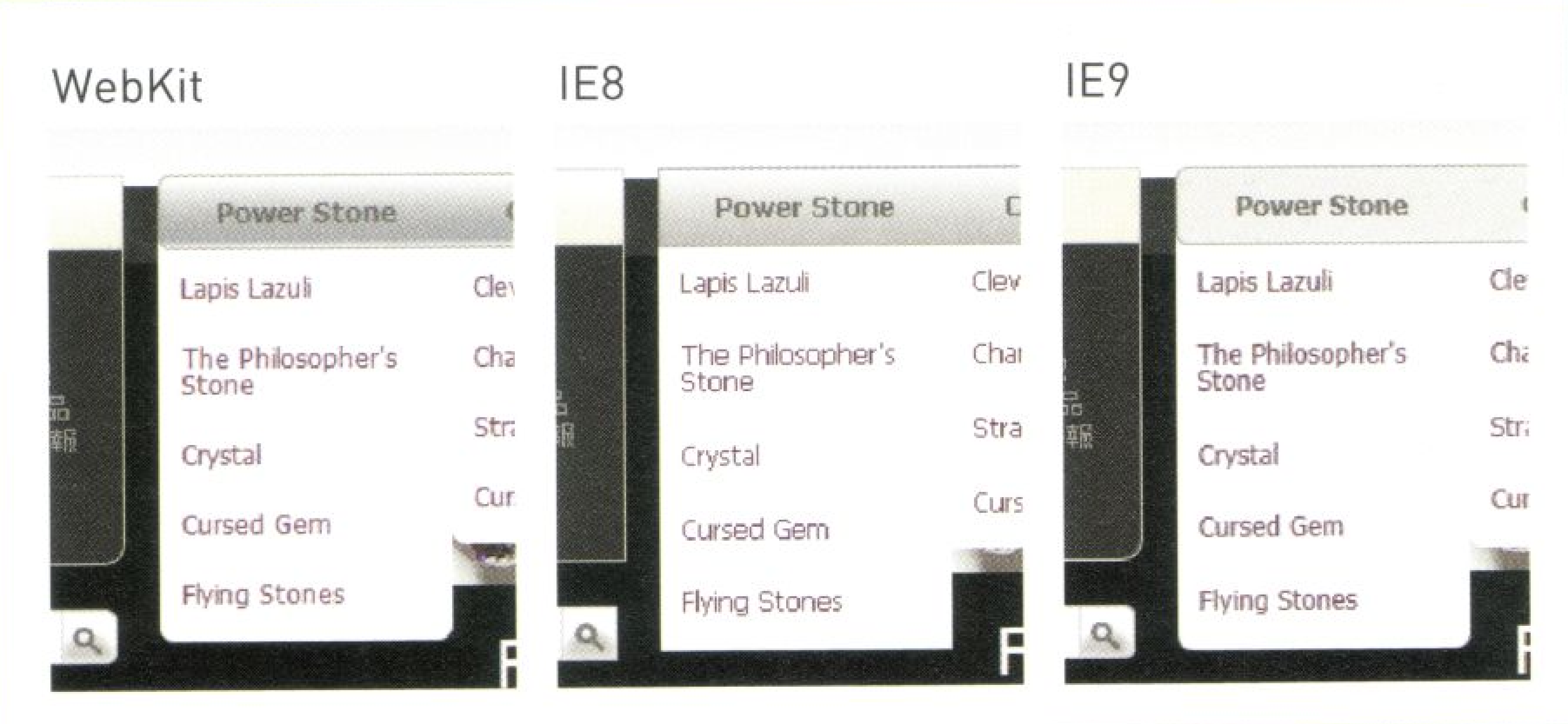
```
#mainContent ul#itemMenu li ul.sub li
a:hover {
background-image:none;
background-color: #EBDEE7;
border: none;
}
```

Internet Explorerでの表示を確認

10

IE8・IE9でのデザインはCSS3の対応がまだまだなので、WebKit系やFirefoxと比べると完全な表現になっていません 10-1。
IE8ではグラデーションをfilterを利用して表現していますが、角丸は再現できません。
IE9ではグラデーションの表現をあきらめた代わりに、角丸が表示された状態になっています。グラデーション部分はベタ塗りになっています。

10-1 WebKit、IE8、IE9での表示の違い



11

IE8用のfilterでグラデーションを表示するには、条件分岐を利用して専用スタイルシートをリンクさせます 11-1。
IE9用にグラデーションの代わりにベタ塗りをする手法は、ほかのブラウザ用にグラデーションを指定している箇所の後に、グラデーションの中でメインになる色をbackground-colorで指定しています。

11-1 IE8対応の条件分岐

```
<!--[if lte IE 8]>
<link rel="stylesheet" href="ie8.css" type="text/css" />
<![endif]-->
```

TIPS

IEの条件分岐はCSSやJavaScriptのIE対策でよく利用されていますが、headタグ内限定の機能ではなく、body内部にも使うことができます。
IE7以下を使っている人にブラウザの変更を促す表示をすることも、body内に 01 のように書くだけで可能です。JavaScriptを利用しなくても簡単にできますね。

01 IEの条件分岐のbody内での使用例

```
<!--[if lte IE 7]><div style="font-size: 500%;">そのWebブラウザは危険です！安全な
Chromeに乗り換えよう！！</div><![endif]-->
```

プルダウンの動きを表現する

12

まずは、通常時は子メニューが非表示の状態で、親メニューにロールオーバーすると子メニューが表示されるようにしてみます 12-1 。
通常時のulにdisplay:noneを付けて、親が:hover状態のときにdisplay:blockすれば、子メニューが「さっ」と表示される軽快な動きを表現できます。

12-1 子メニューを表示させる

```
#mainContent ul#itemMenu li ul.sub {
display:none;
position:absolute;
z-index: 10;
font-size: 10px;
top: 30px;
padding-top: 5px;
padding-bottom: 6px;
background: #FFF;
border-radius:0 0 6px 6px;
}
#mainContent ul#itemMenu li:hover
ul.sub {
display:block;
}
```

13

子メニューがプルダウンして表示される際のモーションを、さらに追加していきます。
ここでは、現在WebKit系のみの対応となりますが、transition効果を使って、子メニューが少しゆっくりと降りてくる動きを表現します 13-1 。
WebKit系以外のブラウザでは、transitionによる子メニューがゆっくり現れる効果が表現できないため、普通の動きになります。

13-1 transitionの設定

```
-webkit-transition: opacity 200ms linear, top 200ms ease-in;
```

- 動きが完了するまでの時間。ミリ秒で指定。単位をsにすると秒指定になる。
- 加速度の付き方
- 変化させたいプロパティ
- コンマ区切りで連続して書くことも可能

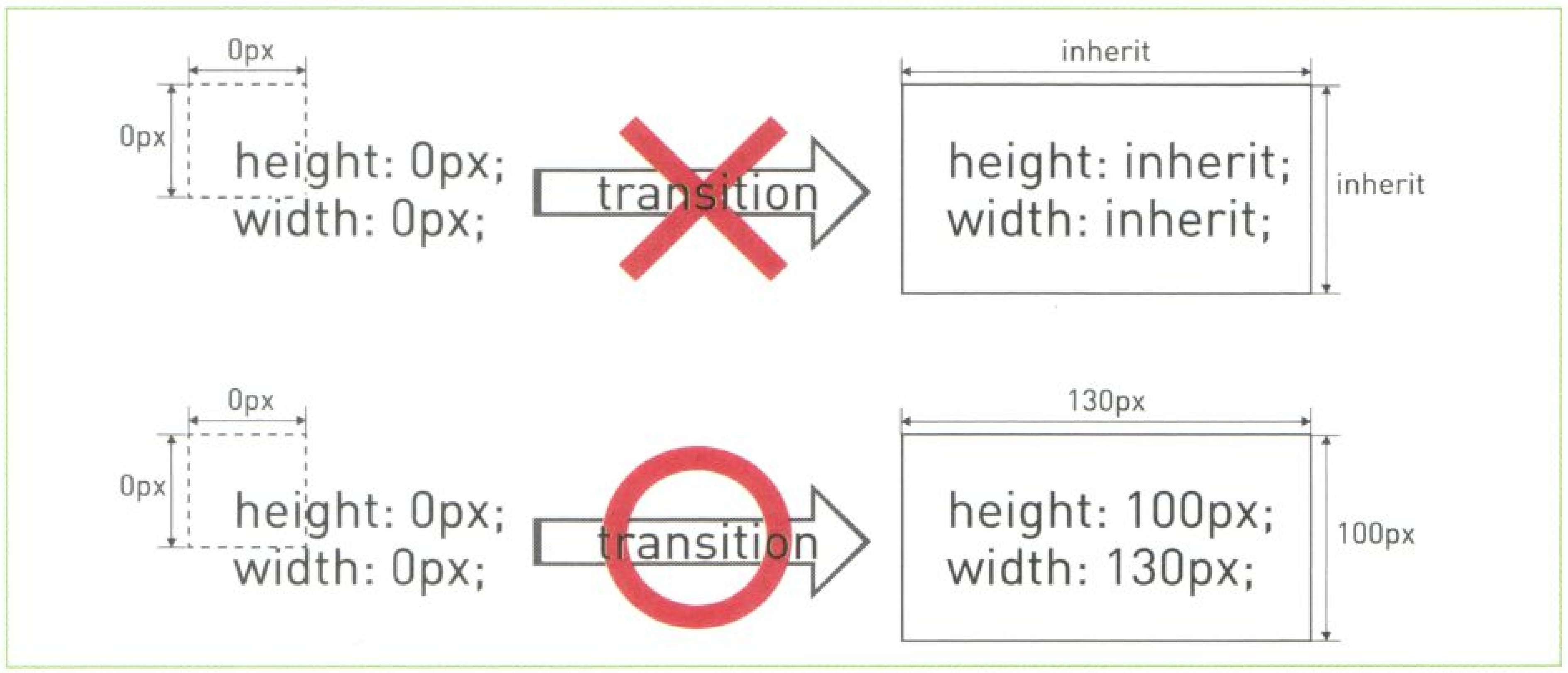
14

残念なことにtransitionでは、いくつか扱えるプロパティに制限があります。displayプロパティにはtransitionを使って動きをつけることができなかったため、12で設定したul要素をdisplay:none→display:blockとして、ulの:hover時に子メニューを表示することができなくなりました。
また、ここではメニュー項目の増減に柔軟に対応できる作りにしたいため、heightの値は固定した数値で指定せず、height:inheritとしています。transitionを使って、widthとheightのサイズを変更することはできるのですが、height:inheritのように数値を指定しない値ではtransitionが適用されません 14-1 。

> **MEMO**
>
> inheritは親要素のプロパティ値を、そのまま子要素に継承することを表す値です。inheritを使うことで、通常は親要素から継承しないプロパティの値を継承させることができます。



14-1 transition しない設定と、する設定



15

このためdisplay:none以外の方法で、ulの通常時に子メニューを隠しておくことにします。ここでは、opacityとmarginを、transitionに利用することにしました。
また、通常時の子メニューを隠すために、widthとheightを0→:hover時にinheritへ切り換えを行うことにします **15-1**。
heightの値の調整は、今回の例だとtransitionには使えませんでしたが、表示・非表示の仕掛けには利用できました。

15-1 opacityとmariginを利用する

```
#mainContent ul#itemMenu li ul.sub {
height: 0;
width: 0;
overflow:hidden;
— 省略 —
top: 0;
— 省略 —
opacity: 0.5;
— 省略 —
}
```

16

-webkit-transitionの値は、カンマ区切りで複数指定できます。
変更したいプロパティと、変化にかかる時間・加速度の付き方を設定しました **16-1**。
これでできあがりです。

16-1 -webkit-transitionの値の設定

```
#mainContent ul#itemMenu li ul.sub {
— 省略 —
-webkit-transition: opacity 200ms linear, top 200ms ease-in;
}
```

まとめ [CONCLUSION]

transitionを使った表現は現時点ではWebKitしか動きませんが、CSS3のプロパティのひとつなので、ほかのブラウザも徐々に対応していくでしょう。
プルダウンメニューをWebKit系に対応させる必要がないのであれば、13以降のtransitionを使った部分は実装しなくていいので、もっと早く仕上がるでしょう。
メニューをデザイン通りにスタイリングするのは、地味で手間がかかります。はじめは単純なデザインからやってみましょう。

02 表示内容を切り替えるタブナビゲーションの実装



CSS3から追加された擬似クラス:targetを利用すれば、JavaScriptを使わずともナビゲーションを操作することが可能になります。
タブボタン形式のシンプルなフォトギャラリーをCSSで実装してみましょう。

制作・文／山本圭助
（株式会社ラナデザインアソシエイツ）

制作のポイント

- コンテンツの重ね方
- 擬似クラス:targetの使い方
- 角丸タブの作り方

対応ブラウザ

IE	P.E.	Safari	5以上
Firefox	8以上	Chrome	15以上

※JavaScript使用時はIE6から対応（一部の効果には非対応）。

TECHNIC 1

タブとコンテンツをセットで扱う

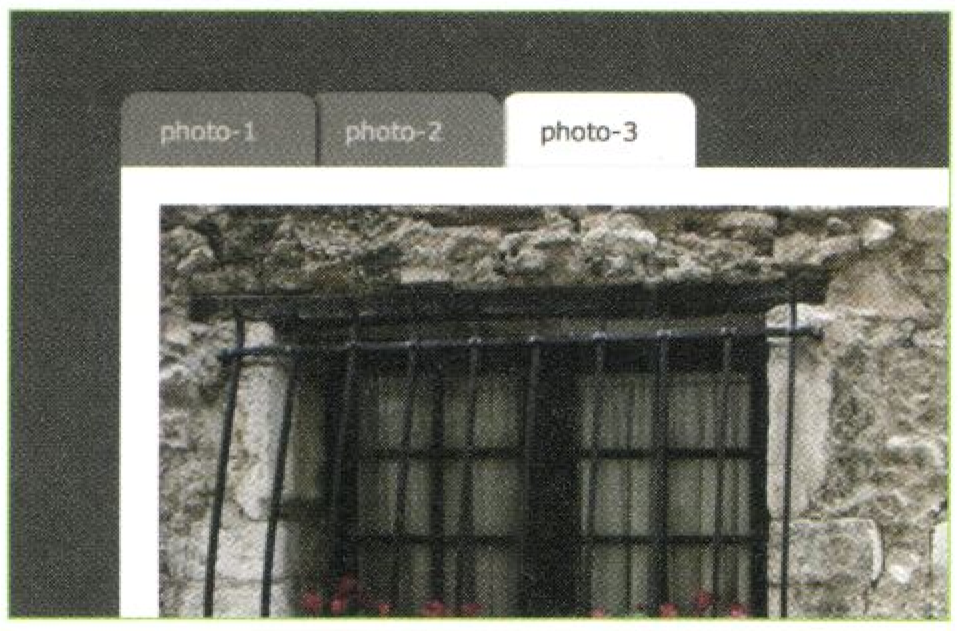
タブとコンテンツをセットして1つのブロックとして扱うことで、それぞれの重なりと位置をまとめて指定できます。

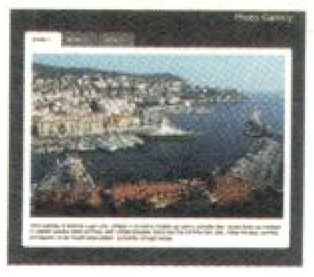


TECHNIC 2

フェード効果を有効に使う

タブへのマウスオーバー時とコンテンツの切り替え時にそれぞれフェード処理を施し、効果的に演出します。





表示する複数のコンテンツを重ねる

01

タブとコンテンツを1セットにしたブロックを3つ用意します 01-1 。
今回は、dl要素（クラス名.page）をコンテナとし、その中のdt要素（クラス名.tab)をタブとして、dd要素（クラス名.content）をコンテンツエリアとして使用します。
タブにあたるdt要素の中にa要素を作成し、タブを内包するブロックへリンクするようにしておきます。

01-1 タブとコンテンツのセット

```
<body>

  <dl id="page-1" class="page">
    <dt class="tab">
      <a href="#page-1">page-1 タブ</a>
    </dt>
    <dd class="content">
      page 1 コンテンツ
    </dd>
  </dl>

  <dl id="page-2" class="page">
    <dt class="tab">
      <a href="#page-2">page-2 タブ</a>
    </dt>
    <dd class="content">
      page 2 コンテンツ
    </dd>
  </dl>

  <dl id="page-3" class="page">
    ― 省略 ―
  </dl>

</body>
```

02

ブロック（.page）のpositionをabsoluteにし、同じ場所に重ねます 02-1 。
タブ（.tab）とコンテンツ（.content）も同様に指定します 02-2 。

02-1 ブロックを同じ場所に重ねる

```
.page {
  position: absolute;
}
```

02-2 タブとコンテンツを同じ場所に重ねる

```
.tab {
  position: absolute;
  width: 100px;
  height: 40px;
  background-color: #666;
}

.content {
  position: absolute;
  top: 40px;
  width: 460px;
  padding: 20px;
}
```

03

各ブロックごとのタブの位置をleftで、コンテンツの重なりをz-indexで指定します 03-1 。
一番左のタブが一番上になるように、値を指定します。

03-1 タブの位置とコンテンツの重なりを指定する

```
#page-1 .tab {
  left: 0;
  z-index: 3;
}
#page-1 .content {
  z-index: 13;
}

#page-2 .tab {
  left: 95px;
  z-index: 2;
}
#page-2 .content {
  z-index: 12;
}

#page-3 .tab {
  left: 195px;
  z-index: 1;
}
#page-3 .content {
  z-index: 11;
}
```

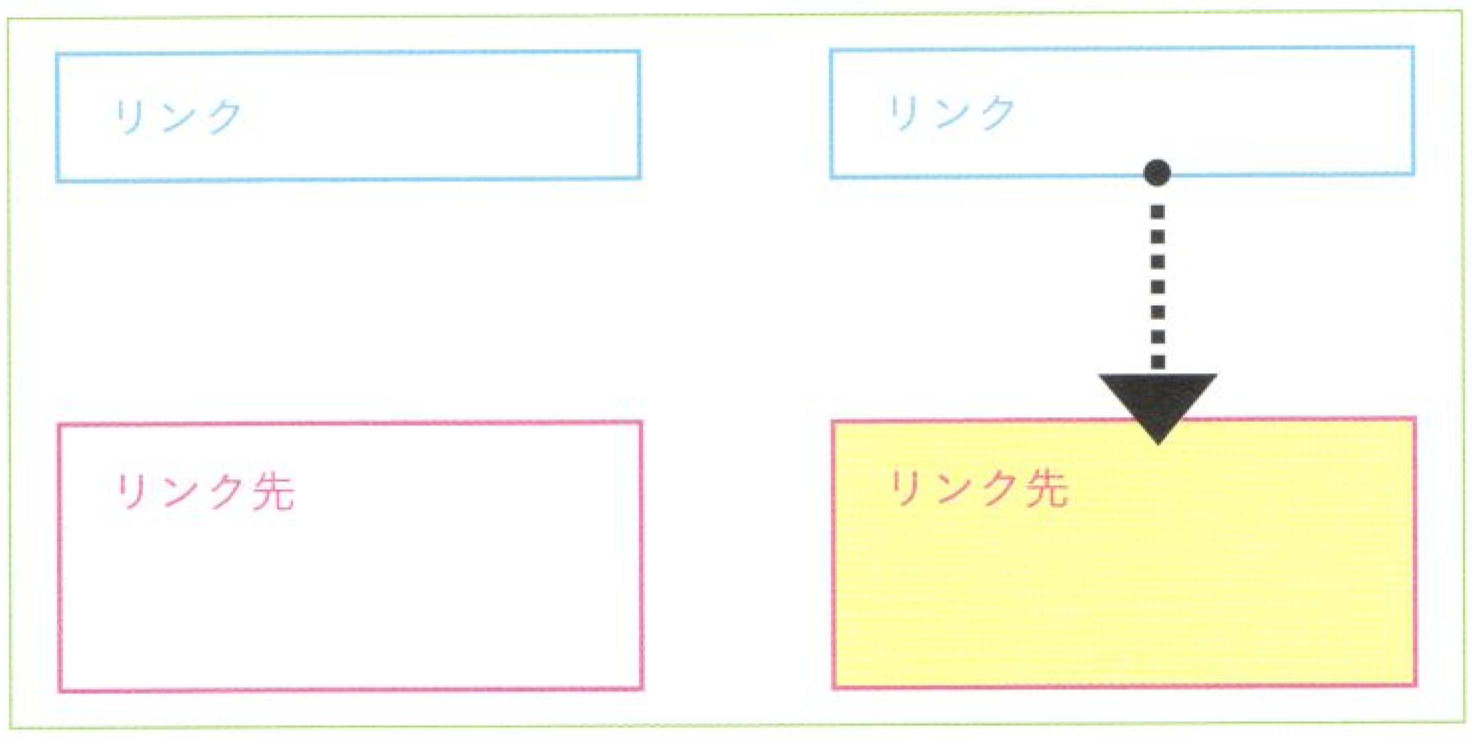
擬似クラス :targetの使い方

04

擬似クラス :targetとは、リンク先に指定された要素が、リンクから呼び出されたとき（ターゲットにされたとき）のスタイルを定義することができる擬似クラスです 04-1。

04-2 の例では、リンクをクリックすると、リンク先であるdiv要素の背景色が変更されます。

04-1 :targetのイメージ



04-2 リンクをクリックするとリンク先の背景色が変わる

```
// html
<a href="#box">リンク</a>

<div id="box">リンク先</div>

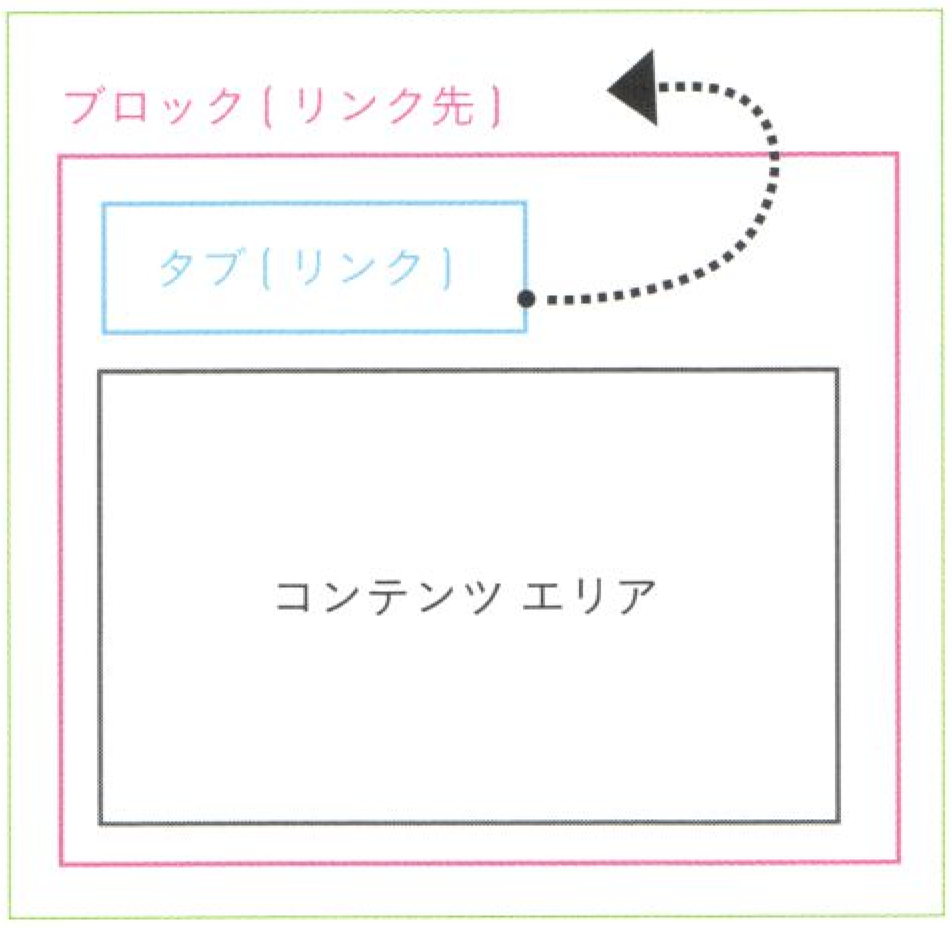
// css
#box {
  background-color: #fff;
}
#box:target {
  background-color: #999;
}
```

05

タブ（.tab）のターゲット先を自身を内包するブロック（.page）にすることで、タブも含めてスタイルを変更できるようになります 05-1。

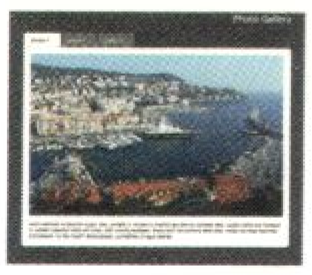
05-2 のように :targetを指定すると、選択したタブのコンテンツが一番手前に表示されるようになります。

05-1 タブも含めたスタイル変更のイメージ



05-2 選択したタブのコンテンツを一番手前に表示する

```
.page:target .tab {
  background-color: #ccc;
  z-index: 1;
}
.page:target .tab a {
  color: #000;
}
.page:target .content {
  z-index: 1;
}
```



角丸タブを実装する

06

タブの見栄えをそれらしくするように、上端を角丸にします 06-1 06-2。

06-1 タブを角丸にする

```
.tab {
  border-radius: 10px 10px 0 0;
}
```

06-2 角丸の指定



*COLUMN

border-radiusプロパティの指定

border-radiusを使って、要素の角丸を指定することができます。
「border-radius: 左上 右上 右下 左下;」で4か所それぞれを指定しますが、省略することで一括指定することもできます 01。値の指定は、px以外にもemや%なども使用できます。

01 border-radiusの省略指定の例

記述	意味
border-radius: 5px;	すべての角を半径5pxのカーブに指定
border-radius: 4px 8px;	左上と右下を4px、右上と左下を8pxに指定

07

タブの重なりがよりわかりやすくなるように、影を付けます 07-1 07-2。

07-1 タブに影を付ける

```
.tab {
  box-shadow: 2px 0 2px rgba(0, 0, 0, 0.75);
}
```

07-2 タブの影



*COLUMN

box-shadowプロパティの指定

box-shadowを使うと、要素の影を指定することができます。
「box-shadow: 水平方向の距離 垂直方向の距離 ぼかし幅 拡散幅 色 向き;」で各値を指定します 01。水平距離と垂直距離以外の値は省略可能です。
向きにinsetを指定すると、要素の内向きに影が付きます。

01 box-shadowの記述例

記述	意味
box-shadow: 5px 10px 3px #ccc;	要素に対して、右に5px、下に10px、ぼかし幅3px、色#cccの影を付けます

タブ切り替えのアニメーション

08

タブ切り替え時のアニメーションを加えます。
transition-durationで時間を指定すると、値が変更される際にアニメーションするようになります。タブとコンテンツエリアに、それぞれtransition-durationを指定します。
transitionプロパティ関係は、まだベンダープレフィックスが必要です。 08-1

08-1 タブとコンテンツエリアにtransition-durationを指定

```
.tab {
  -webkit-transition-duration: 400ms;
  -moz-transition-duration: 400ms;
}
.content {
  -webkit-transition-duration: 400ms;
  -moz-transition-duration: 400ms;
}
```

ATTENTION!

CSS3はまだ最終勧告ではないため、一部のプロパティはブラウザごとにベンダープレフィックスを付けて指定する必要があります 01。

01 ブラウザごとのベンダープレフィックス

ブラウザ	ベンダープレフィックス
Firefox	-moz-
Safari、Chrome	-webkit-
Opera	-o-
Internet Explorer	-ms-

*COLUMN

transitionプロパティの指定

transitionは、要素の値を変更する際のアニメーション（トランジション）を指定することができます 01。

01 transitionプロパティの要素

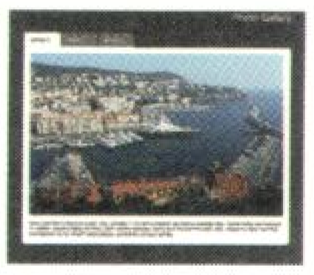
要素	機能
transition-property	適用するプロパティ名を指定することで、アニメーションするプロパティを限定することができます。カンマ区切りで複数指定も可能です
transition-duration	アニメーションにかける時間を指定します。単位はs（秒）かms（ミリ秒）で、0の場合でも単位は必要です。 カンマ区切りで複数指定することで、transition-propertyで指定した複数プロパティそれぞれの時間を指定することができます。 transition-propertyで指定したプロパティ数より少ない場合は、繰り返しの扱いになります。02は03と同じコードになります
transition-timing-function	アニメーションの加速や減速などのイージングを指定します。 easeがデフォルトで、他にlinear、ease-in、ease-out、ease-in-outがあります
transition-delay	アニメーションが開始されるまでの待ち時間を指定します

02 transition-durationの時間指定が、transition-propertyで指定したプロパティ数より少ない

```
transition-property: opacity, left, top, height;
transition-duration: 3s, 5s;
```

03 transition-durationの時間指定が、transition-propertyで指定したプロパティ数と同じ

```
transition-property: opacity, left, top, height;
transition-duration: 3s, 5s, 3s, 5s;
```



09

コンテンツエリアにopacityを指定することで、タブ選択時にコンテンツエリアにフェード効果を加えます 09-1 。

09-1 コンテンツエリアにフェード効果を加える

```
.content {
  opacity: 0;
}
.page:target .content {
  opacity: 1;
}
```

Internet Explorerへの対応

10

IE8以下では擬似クラス:targetに対応していませんので、JavaScriptで代用します。
10-1 の例では、jQuery 10-2 を使用しています。

10-1 IE向けにJavaScriptで代用

```
$(function() {
  $(".page").each(function() {
    var _this = this;
    $(".tab a", this).click(function() {
      $(".content").hide();
      $(".content", _this).show();
      return false;
    });
  });
});
```

10-2 jQuery

http://jquery.com/

TIPS

IEでもCSS3のセレクタが利用できるようにする「selectivizr.js」 01 というスクリプトもあります。HTMLのhead要素部分を 02 のようにすれば利用できます。

01 selectivizr.js

http://selectivizr.com/

01 selectivizr.js

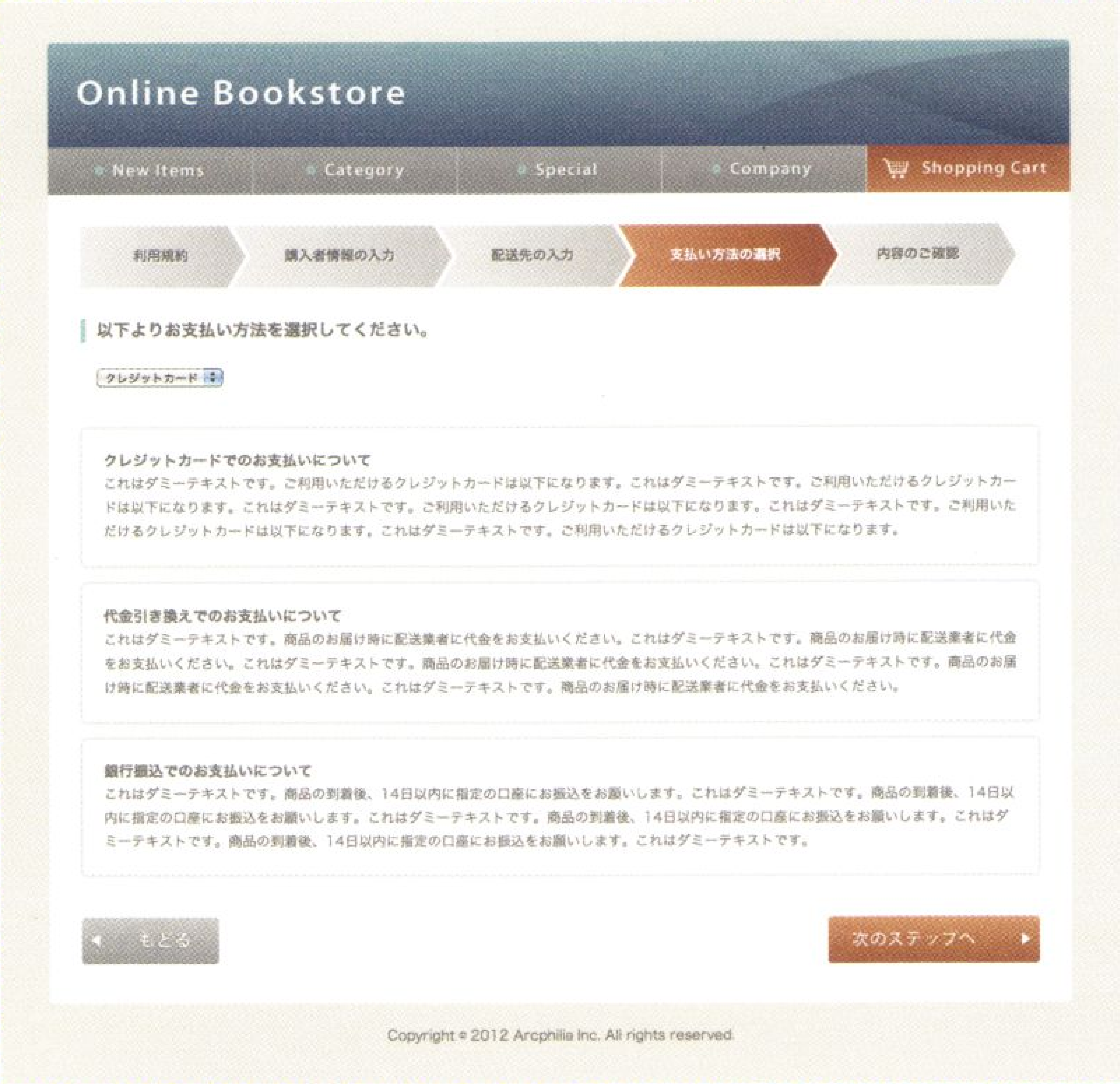
```
<!--[if lte IE 8]>
<script type="text/javascript" src="js/selectivizr.js"></script>
<![endif]-->
```

まとめ [CONCLUSION]

擬似クラス:targetはリンクと組み合わせることで、より多様性のあるナビゲーションやコンテンツを手軽に実装することができます。

ただしIE8以下は非対応なので、スタイルシートの使い分けやJavaScriptの使用などでコンテンツ自体が機能しなくなることを防ぎましょう。

03 画像を使わないで作るパンくず型の購入フロー



ECサイトの「ショッピングカート」内などでよく見られる、パンくず型の購入フローを制作します。フローの画像を全ページ分用意すると面倒ですが、CSS3を利用することで、画像を使わずに美しいフローを完成させることができます。

制作・文／高村香奈子（株式会社アークフィリア）

制作のポイント

- transform:skewで平行四辺形を描画する
- ::beforeと::afterを利用してパーツを組み合わせる
- a要素に対してロールオーバー時の効果を指定する

対応ブラウザ

IE	P.E.	Safari	5以上
Firefox	8以上	Chrome	16以上

TECHNIC 1

同色系のグラデーションを使用した表現

なめらかな見た目になるように、同色系のグラデーションを使用しています。



TECHNIC 2

フロー内の現在地をわかりやすく可視化

現在地のみ色を変え、現在地をわかりやすくしています。





フロー部分のベースを作る

01

フロー部分のHTMLはulで記述します。フローの左に表示させたい項目から順に記述し、ulにはクラスbreadを、現在地となる項目にはクラスcurrentを付与しておきます 01-1 01-2 。

01-1 フロー部分のHTML

```
<ul class="bread">
  <li><a href="#">利用規約</a></li>
  <li><a href="#">購入者情報の入力</a></li>
  <li><a href="#">配送先の入力</a></li>
  <li class="current"><a href="#">支払い方法の選択</a></li>
  <li><a href="#">内容のご確認</a></li>
</ul>
```

> **MEMO**
>
> ここでのDOCTYPEはHTML5.0を使用しています。

01-2 ulの表示



02

liをfloat:left;で横並びにし、高さを60pxに指定しておきます 02-1 02-2 。フローの三角の部分は2つの図形を組み合わせて作るため、高さの数値は偶数にしてください（後のステップで詳しく解説します）。

02-1 ベースのレイアウト

```
.bread {
  list-style: none;
  overflow: hidden;
}
.bread li {
  float: left;
  line-height: 60px;
}
```

> **MEMO**
>
> 今回は、HTML5のリセット・スタイルシート「HTML5 Reset Stylesheet」01を別途読み込ませています。
>
> **01 HTML5 Reset Stylesheet**
>
> http://html5doctor.com/html-5-reset-stylesheet/

02-2 liを横並びにしたところ



03

aをブロック要素にし、paddingや文字のスタイルを指定します03-1。
次に、背景のgradientとtext-shadowの指定をし、ディテールを作り込んでいきます。
gradientは、#EDEDEDから#CCCCCCの薄いグレーのグラデーションに設定します。また、それぞれのブラウザ用にベンダープレフィックスを付けるほか、gradient未対応のブラウザに備えてbackground-colorも併記しておきます。
text-shadowは、エンボスのような効果を出すため、1pxの白いシャドウを付けました03-2。

03-1 a要素のスタイル

```
.bread li a {
  display: block;
  float: left;
  padding: 0 40px 0 50px;
  font-weight:bold;
  color: #666666;
  text-decoration: none;
  /* CSS3 */
  background: -moz-linear-gradient(left, #EDEDED, #CCCCCC);
  background: -webkit-linear-gradient(left, #EDEDED, #CCCCCC);
  background: -ms-linear-gradient(left, #EDEDED, #CCCCCC);
  background: linear-gradient(left, #EDEDED, #CCCCCC);
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#EDEDED', endColor
  str='#CCCCCC',GradientType=1 );
  background-color:#CCCCCC;
  text-shadow: 1px 1px 0 #FFFFFF;
}
```

03-2 グラデーションなどを指定したところ

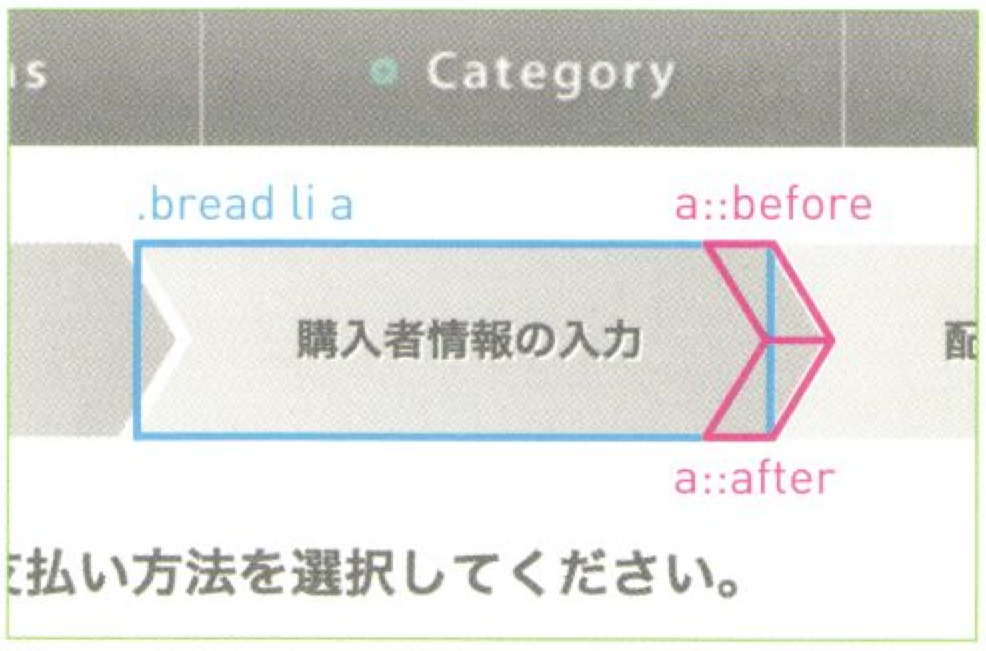


三角形の部分を描く

04

続いて、フロー端の三角の部分を描いていきます。
aの擬似要素::beforeと::afterに対し、要素を傾斜変形させるtransform:skewを利用して2つの平行四辺形を描き、これらを組み合わせて三角のかたちをつくります04-1。

04-1 フローをつくるためのパーツの構造





05

擬似要素::beforeをブロック要素にし、transform:skew(30deg);を設定して、傾斜角度が30度の平行四辺形を作ります。aのボックスに対して上半分に配置するため、マイナスマージンを設定し、位置を合わせます。また、隣の項目との余白をとるために、#FFFFFFのborderを設定しました 05-1 05-2。

> **MEMO**
>
> 擬似要素の高さは、liに指定した高さ60pxに対して半分の30pxに設定します。

05-1 a::beforeのスタイル

```
.bread li a::before {
  display: block;
  width: 15px;
  height: 30px;
  border-right: 5px solid #FFFFFF;
  margin:0 -50px -30px auto;
  background-color: #CCCCCC;
  /* CSS3 */
  content: " ";
  -webkit-transform: skew(30deg);
  -moz-transform: skew(30deg);
  transform: skew(30deg);
}
```

05-2 上半分の平行四辺形



06

05と同様に、擬似要素::afterのスタイルを設定していきます。05の平行四辺形と逆のかたちになるように、transform:skew(-30deg);を設定して、傾斜角度がマイナス30度の平行四辺形を作ります。また、a要素のボックスに対して下半分に配置するため、こちらもマイナスマージンを設定し、位置を調整します 06-1 06-2。

06-1 a::afterのスタイル

```
.bread li a::after {
  display: block;
  width: 15px;
  height: 30px;
  border-right: 5px solid #FFFFFF;
  margin:-30px -50px 0 auto;
  background-color: #CCCCCC;
  /* CSS3 */
  content: " ";
  -webkit-transform: skew(-30deg);
  -moz-transform: skew(-30deg);
  transform: skew(-30deg);
}
```

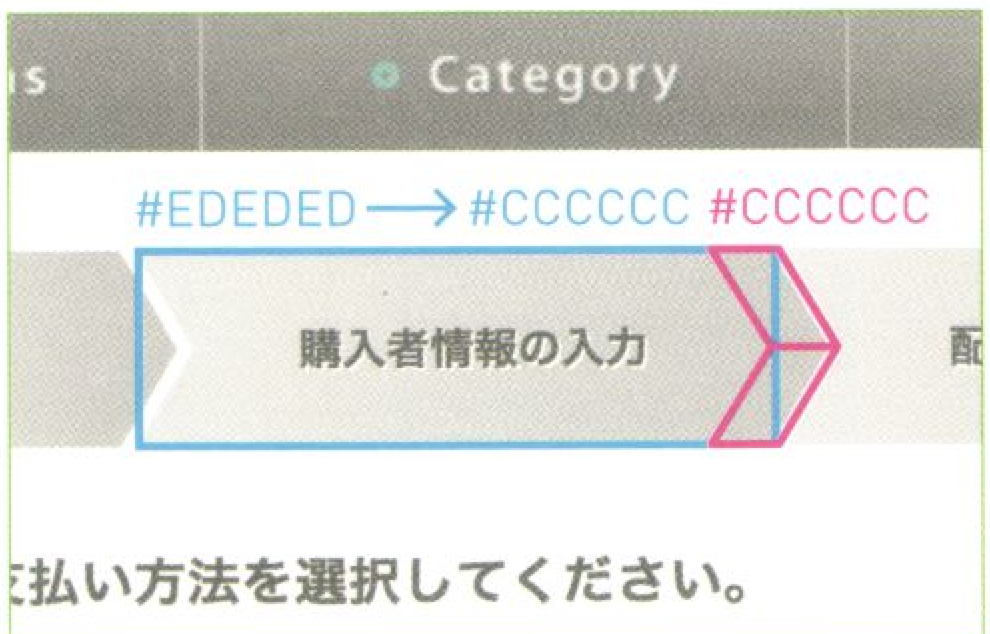
06-2 下半分の平行四辺形



07

以上でフローの三角の部分ができました。上下それぞれの平行四辺形の背景色は、a要素のgradientの終了色#CCCCCCに指定することで、a要素のボックスとつながったグラデーションのように見せていることがポイントです 07-1。

07-1 gradientとbackground-color



現在地の表現を工夫する

08

次に、クラスcurrentを付けておいた現在地の表現をわかりやすくするため、背景色や文字色などをオレンジ色ベースに変更していきます 08-1。

08-1 現在地の表現



09

文字色を#FFFFFF、gradientを#E69A4Fから#BA4A0Aの線形のグラデーションに設定します 09-1。text-shadowは、背景のグラデーションよりも少し濃いオレンジ色に指定しました 09-2。

09-2 オレンジ色のグラデーション



09-1 現在地のスタイル

```
.bread li.current a {
  color: #FFFFFF;
  /* CSS3 */
  background: -moz-linear-gradient(left, #E69A4F, #BA4A0A);
  background: -webkit-linear-gradient(left, #E69A4F, #BA4A0A);
  background: -ms-linear-gradient(left, #E69A4F, #BA4A0A);
  background: linear-gradient(left, #E69A4F, #BA4A0A);
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#E69A4F', endColorstr='#BA4A0A',GradientType=1 );
  background-color:#BA4A0A;
  text-shadow: 1px 1px 0 #BA4A0A;
```

10

三角の部分がグレーのままなので、gradientの終了色である#BA4A0Aに指定し、a要素のボックスとつながったグラデーションのような処理にします 10-1 10-2。

10-1 a::beforeとa::afterの背景色

```
.bread li.current a::before,
.bread li.current a::after {
  background-color: #BA4A0A;
}
```

10-2 三角の部分の背景色





ロールオーバー時のスタイルを設定する

11

続いて、フローの各項目をロールオーバーした時のスタイルを設定していきます。画像を使用していないので、gradientの値を変えるだけで簡単にロールオーバー時の効果を与えることができます 11-1。

11-1 ロールオーバー時



12

現在地以外の項目に対しては、#CCCCCCから#999999のやや濃いグレーのグラデーションを設定します 12-1。背景色が濃くなったので文字色を#FFFFFFにし、text-shadowもやや濃いグレーの#999999に指定しました 12-2。

12-2 ロールオーバー時のグラデーション



12-1 ロールオーバー時のスタイル

```
.bread li a:hover {
  color: #FFFFFF;
  /* CSS3 */
  background: -moz-linear-gradient(left, #CCCCCC, #999999);
  background: -webkit-linear-gradient(left, #CCCCCC, #999999);
  background: -ms-linear-gradient(left, #CCCCCC, #999999);
  background: linear-gradient(left, #CCCCCC, #999999);
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#CCCCCC', endColorstr='#999999',GradientType=1 );
  background-color:#999999;
  text-shadow: 1px 1px 0 #999999;
}
```

13

三角の部分がロールオーバー前の色のままなので、::beforeと::afterに関しても、背景色をやや濃いグレーの#999999に指定しておきます 13-1 13-2。

13-1 ロールオーバー時の背景色

```
.bread li a:hover::before,
.bread li a:hover::after {
  background: #999999;
}
```

13-2 三角の部分の背景色



14

現在地のロールオーバー時に関しても、同じように#CB6925から#A03200のやや濃いオレンジ色のグラデーションに設定します。文字色は白のままですが、text-shadowをやや濃いオレンジ色に指定しました 14-1 14-2。

14-2 ロールオーバー時のグラデーション



14-1 ロールオーバー時のスタイル

```
.bread li.current a:hover {
  /* CSS3 */
  background: -moz-linear-gradient(left, #CB6925, #A03200);
  background: -webkit-linear-gradient(left, #CB6925, #A03200);
  background: -ms-linear-gradient(left, #CB6925, #A03200);
  background: linear-gradient(left, #CB6925, #A03200);
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#CB6925', endColorstr='#A03200',GradientType=1 );
  background-color:#A03200;
  text-shadow: 1px 1px 0 #A03200;
}
```

15

こちらも13と同じく、三角の部分がロールオーバー前の色のままなので、::beforeと::afterに関しても、背景色を#A03200に指定しておきます 15-1 15-2。

15-1 ロールオーバー時の背景色

```
.bread li.current a:hover::before,
.bread li.current a:hover::after {
  background: #A03200;
```

15-2 三角の部分の背景色



16

以上で購入フローの完成です。このほかのページでは、クラスcurrentの位置を付け替えることで、現在地を簡単に移動させることができます 16-1。

16-1 クラスcurrentを移動したところ





*COLUMN

最低限のユーザビリティを保つには

本作例は、プログレッシブ・エンハンスメントの考えをもとに制作しています。例えば、IEではtransform:skewが未対応のため、三角の部分の描画はできませんが、グラデーション部分にfilter（IE独自の視覚効果を与えるプロパティ）を併記することにより、IE6～8でも再現することが可能です。このため、三角の部分が見えなくても、項目の区切りは認識することができます 01 02。
また、gradientだけでなく、念のためbackground-colorも併記しておくことにより、グラデーション未対応のブラウザで見た際もフローの形状は認識することができますので、最低限のユーザビリティは保たれます。
IEを含めたすべてのブラウザで三角の部分を描画する必要がある場合は、別途画像にするか、JavaScriptの使用などを検討してください。

01 filterを記述し、IE8で閲覧したところ



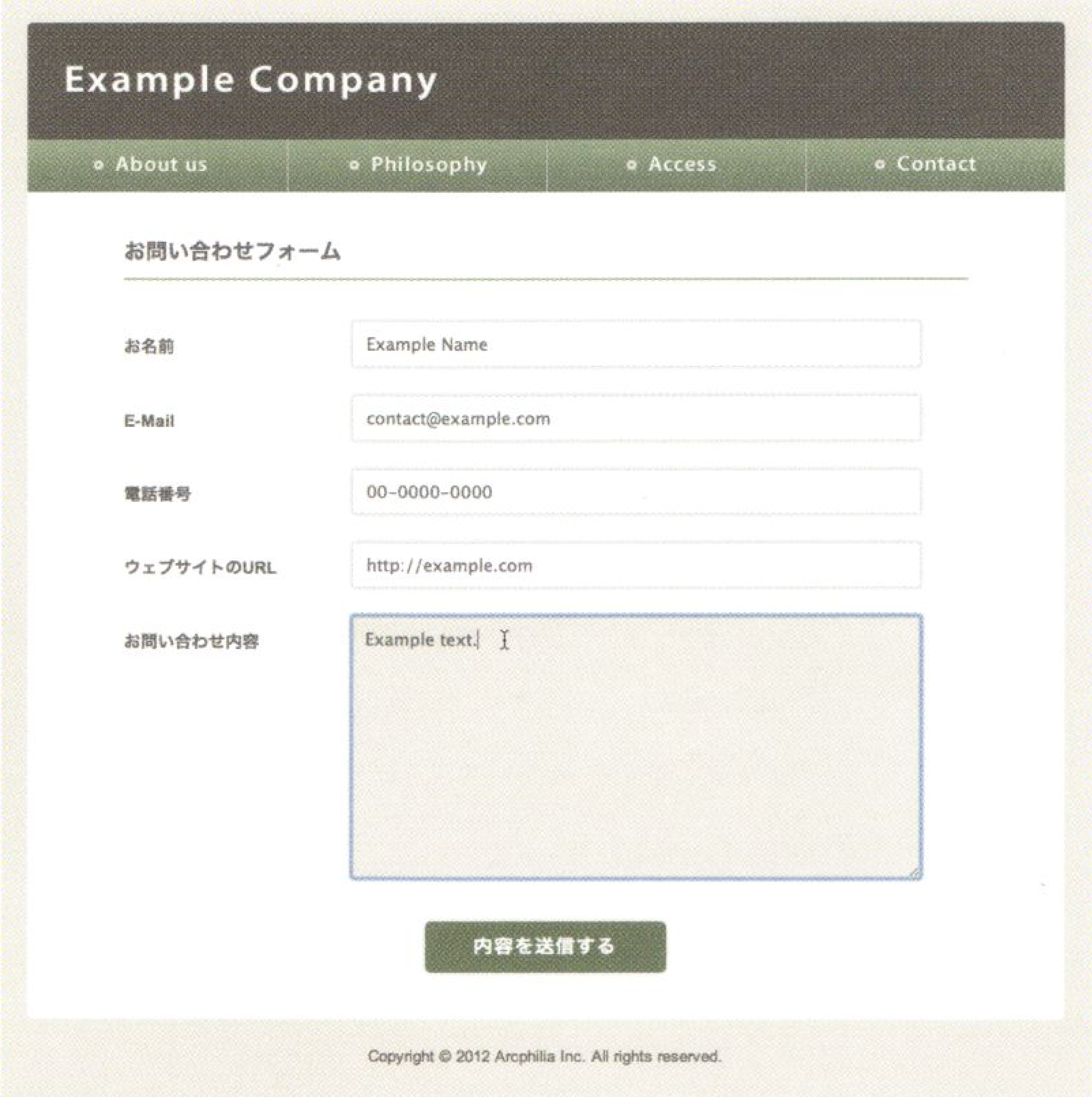
02 IE6～8でfilter未記述の場合、項目の区切りがわからなくなってしまう



まとめ [CONCLUSION]

フローをCSSのみで実装することにより、文字数に応じて横幅を自動で伸ばしたり、ロールオーバーの処理も簡単にすることができました。本作例は購入フローを想定して制作していますが、会員登録画面やパンくずリストなど、さまざまな用途で利用することが可能です。
また、今回は現在地以外の項目に対して移動ができるようにリンクを張っていますが、リンクをさせたくない場合は、aではなくliに対して各スタイルを適用するなどの対応をしてください。

04 ユーザビリティに配慮した問い合わせフォーム



テキスト入力欄やボタンにカーソルを合わせると色が変化することで、現在操作している箇所がわかりやすくなるフォームをつくります。CSS3のtransitionを使用して、背景色や影をゆっくりとしたアニメーションで変化させることがポイントです。

制作・文／高村香奈子（株式会社アークフィリア）

制作のポイント

- transitionで背景色や影の色をゆっくり変化させる
- box-shadowを使ったパーツの加工
- border-radiusでボタンを角丸に

対応ブラウザ

IE	P.E.
Safari	5以上
Firefox	8以上
Chrome	16以上

TECHNIC 1

操作箇所のインタラクション

テキスト入力欄にカーソルを合わせると、わずかに背景色が付きます。



TECHNIC 2

送信ボタンのマウスオーバー効果

送信ボタンにカーソルを合わせると、発光しているような効果になります。





フォームのレイアウト

01

DOCTYPEはHTML5.0を使用します。フォームの該当箇所をformタグで囲み、labelとinputをセットで記述します。また「内容を送信する」ボタンには、他のinput要素と区別するため、クラスsubmitを付与しておきます 01-2 01-2。

TIPS

ここで記述したtype="email"、type="tel"、type="url"は、HTML5からtype属性に追加できるようになった値です。例えば、<input type="email">に不正なメールアドレスを入力すると自動でアラートが出るなど、ブラウザ側でそれぞれの値に応じた動作がサポートされています(Google ChromeやFirefoxなど、それぞれの値をサポートしているブラウザに限ります)。

01-1 フォームのHTML

```
<form>
  <label for="name">お名前</label>
  <input type="text" name="name" value="" />
  <label for="email">E-Mail</label>
  <input type="email" name="email" value="" />
  <label for="tel">電話番号</label>
  <input type="tel" name="tel" value="" />
  <label for="url">ウェブサイトのURL</label>
  <input type="url" name="url" value="" />
  <label for="message">お問い合わせ内容</label>
  <textarea name="message" rows="0" cols="0"></textarea>
  <input type="submit" value="内容を送信する" name="submit" class="submit" />
</form>
```

01-2 CSS適用前



02

まず、フォームのlabelとinput、もしくはtextareaを横並びにレイアウトしていきます。それぞれの要素の幅を固定し、labelにfloat:left;を指定して左側に回り込ませることで、横並びにします 02-1。

02-1 各要素の幅



フォームのCSSを記述する

03

それでは実際に、フォーム箇所のCSSを記述していきます。<form>の幅は700pxに固定し、最終的にテキスト入力欄とボタンのレイアウトを中央に揃えるために、text-align:center;を指定しておきます 03-1 03-2。

03-1 formのスタイル

```
form {
  width:700px;
  text-align: center;
}
```

03-2 中央揃えにしたい箇所



MEMO

今回は、HTML5のリセット・スタイルシート「HTML5 Reset Stylesheet」01を別途読み込ませています。

01 HTML5 Reset Stylesheet

http://html5doctor.com/html-5-reset-stylesheet/

04

次に、labelのスタイルを設定します。こちらは幅を150pxに固定して、float:left;で左側に回り込ませます。marginや文字のスタイルも合わせて記述します 04-1。

04-1 labelのスタイル

```
label {
  width: 150px;
  margin-top: 15px;
  float: left;
  font-size: 14px;
  font-weight: bold;
  text-align: left;
}
```

05

inputとtextareaに対して、基本的な見た目のスタイルを適用します 05-1。幅を450pxに固定することで、labelと横並びのレイアウトになりました。また、テキストを入力した際の文字の大きさや色もここで指定します 05-2。

05-1 labelのスタイル

```
input,textarea {
  width: 450px;
  border: 1px #DDDDDD solid;
  margin: 4px 0 20px 0;
  padding: 10px;
  background: #FFFFFF;
  font-size: 14px;
  color: #666666;
```

05-2 文字を入力したところ



MEMO

入力した文字が見やすくなるよう、ある程度のpaddingを指定しておきます。また、次のステップでテキスト入力欄に影を付けるbox-shadowを指定しますが、未対応のブラウザでもユーザビリティを保つために、併せて1pxのborderを指定しています。



影とアニメーション効果を記述する

06

続いて、CSS3を記述していきます。テキスト入力欄の周りに影を付けるためのbox-shadowと、カーソルを合わせた時に背景色をゆっくりと変化させるためのtransitionを指定します 06-1 。

box-shadowの値は、「水平方向の距離 垂直方向の距離 ぼかしの距離 影の色」の順番に記述します。今回は、#CCCCCCの薄いグレーの影を付けました。

transitionの値は、「適用させるプロパティ 変化する秒数 変化の仕方」の順に記述をします。今回は背景色を変化させるので、指定するプロパティは「background」になります。変化の仕方は、開始と終了のタイミングが滑らかになる「ease」を1秒で指定しました 06-2 。

06-1 影とアニメーションの効果

```
    /* CSS3 */
    -webkit-box-shadow: 0 0 6px #CCCCCC;
    -moz-box-shadow: 0 0 6px #CCCCCC;
    box-shadow: 0 0 6px #CCCCCC;
    -webkit-transition: background 1s ease;
    -moz-transition: background 1s ease;
    transition: background 1s ease;
}
```

06-2 ボックスの影



07

テキスト入力欄の見た目が大まかにできてきましたが、「お問い合わせ内容」のtextareaにもう少し高さがほしいので、textarea単体にheightを指定してサイズを調整しておきます 07-1 07-2 。

07-1 textareaの高さ

```
textarea {
    height: 200px;
}
```

07-2 高さを調整したところ



マウスオーバー時の背景色を変化させる

08

inputとtextareaのマウスオーバー時に対して、薄いベージュの背景色#EFEDE9を指定します。入力したテキストが読みづらくならないよう、控えめな色にしましょう 08-1 。

08-1 マウスオーバー時の背景色

```
input:hover,textarea:hover {
    background: #EFEDE9;
}
```

09

以上で、input と textarea にマウスオーバーした際に、背景色がゆっくりと変化するようになりました 09-1。

09-1 背景色の変化



TIPS

今回は transition の変化の仕方に「ease」を指定しましたが、一定の速さになる「linear」、開始がゆっくりになる「ease-in」、終了がゆっくりになる「ease-out」などの値があります。

送信ボタンの形とアニメーション効果を設定する

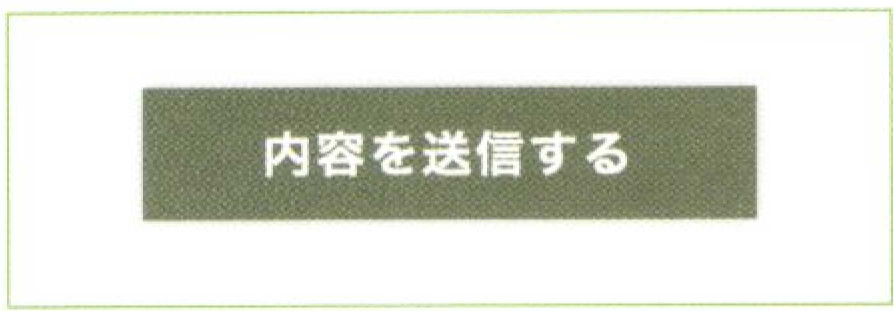
10

続いて、「内容を送信する」ボタンの基本的な見た目のスタイルを設定します。まず input 要素に指定していた回り込みを clear:both; で解除し、ボタンの文字の大きさや色などを指定します。背景色は #78846C のグリーンにしました 10-1 10-2。

10-1 基本のスタイル

```
input.submit {
  clear: both;
  width: 200px;
  border: none;
  margin: 10px auto 0 auto;
  background: #78846C;
  font-size: 16px;
  font-weight: bold;
  color: #FFFFFF;
  letter-spacing:0.1em;
```

10-2 基本のスタイルを適用したところ



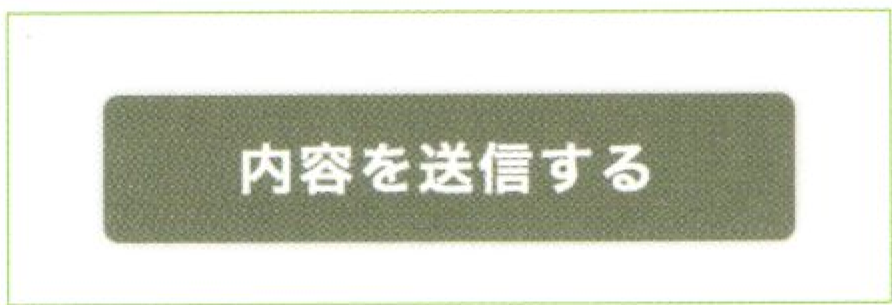
11

CSS3 の border-radius で送信ボタンを角丸の形状にします。四隅の角丸の半径は 5px に指定しました 11-1 11-2。

11-1 角丸の指定

```
  /* CSS3 */
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
```

11-2 送信ボタンの角丸



12

送信ボタンは現在、他の input 要素と同様に #CCCCCC の薄いグレーの影が付いている状態ですが、マウスオーバー時にこの影と背景色を異なる色に変化させましょう。
そのため、アニメーションさせたい箇所は「background」と「box-shadow」の 2 つになりますので、適用させるプロパティには「background,box-shadow」を指定します。このように、カンマで区切ることで特定のプロパティを複数指定することができます 12-1。

TIPS

今回は特定のプロパティを複数指定しましたが、transition の値に「all」を指定すると、すべてのプロパティを対象にすることができます。

12-1 アニメーションの効果

```
  -webkit-transition: background 1s ease, box-shadow 1s ease;
  -moz-transition: background 1s ease, box-shadow 1s ease;
  transition: background 1s ease, box-shadow 1s ease;
}
```



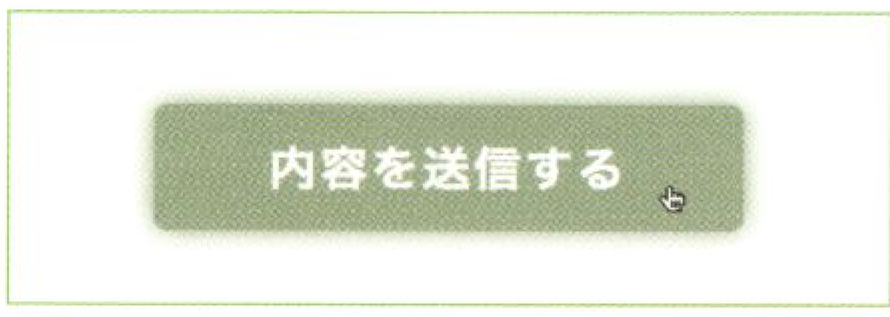
13

マウスオーバー時の送信ボタンに対して、背景色とbox-shadowを#A5B596の薄いグリーンに指定します。box-shadowを背景色と同じグリーンに変化させることで、ボタンが発光しているような効果になりました。ぼかしの距離の値をマウスオーバー前よりも大きめに設定すると、発光の効果をより強く見せることができます。最後に、マウスオーバー時のカーソルの形状がリンクカーソルの形になるよう、pointerを指定します 13-1 13-2。

13-1 マウスオーバー時のスタイル

```
input.submit:hover {
    background: #A5B596;
    -webkit-box-shadow: 0 0 20px #A5B596;
    -moz-box-shadow: 0 0 20px #A5B596;
    box-shadow: 0 0 20px #A5B596;
    cursor: pointer;
}
```

13-2 送信ボタンをマウスオーバーしたところ



14

以上でフォームのスタイルは完成です。本作例はプログレッシブ・エンハンスメントの考え方で実装をしているため、transitionに対応していないIEなどのブラウザで見た場合は、マウスオーバー時のふわっとしたアニメーションは適用されませんが、背景色の変化自体は適用された状態となります。
同様に、box-shadowやborder-radiusの効果に関してもブラウザによっては適用されませんが、borderや背景色の指定によりパーツの形状自体は確認できるため、ユーザビリティは保たれているといえます 14-1。

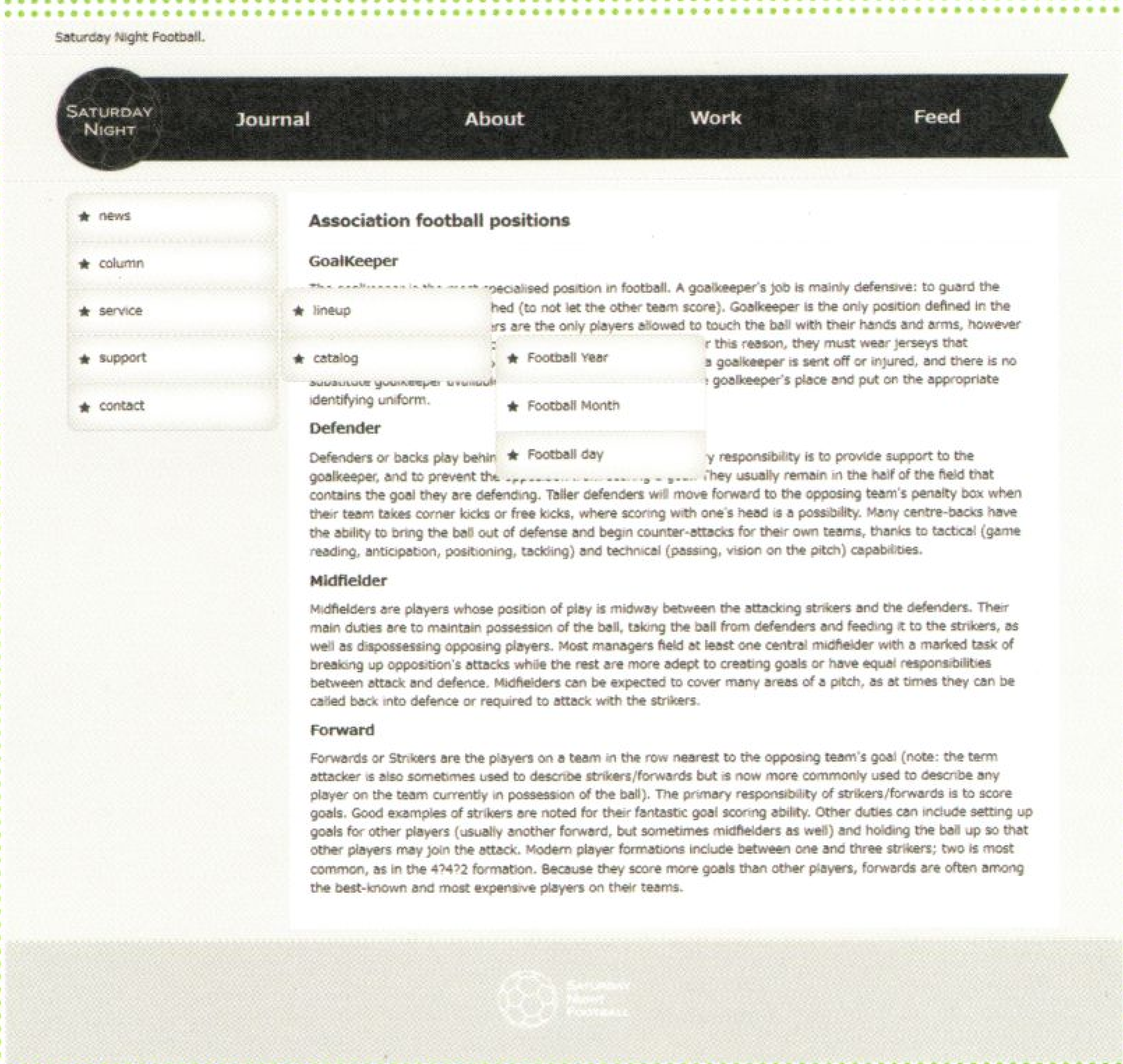
14-1 IE8で表示したところ



まとめ [CONCLUSION]

フォームのインターフェイスにさりげないインタラクションを付けることで、ユーザーが現在操作している箇所がわかりやすくなったり、単調な入力作業の場合でも操作が快適に感じられるようになります。あくまでも入力時の補佐的なインタラクションですので、入力した文字が読みにくくなってしまうことがないように、変化させる背景色などは控えめなものにしましょう。

05 ul要素でシンプルに作る縦並びのプルダウンメニュー



マウスオーバーするとサブメニューが表示される縦並びのプルダウンメニュー。ul要素でシンプルにマークアップすることで、プルダウンの軽快な動きを実現しました。ロールオーバー時のアクションやボタンの装飾についても解説します。

制作・文／中西祐介
（株式会社ワン・トゥー・テン・デザイン）

制作のポイント

- ul要素でシンプルなマークアップ
- li要素全体をリンク範囲に設定
- transition-durationプロパティでロールオーバー時の変化を表現

対応ブラウザ

IE	6以上
Safari	5以上
Firefox	4以上
Chrome	10以上

※IE6～8はbox-shadow、border-radiusに非対応。

TECHNIC 1

CSSがオフの環境化でも可読性を保持

ul要素を入れ子にしたシンプルなマークアップにより、CSSがオフになった環境でもプルダウンメニューがリストとして表示され機能を失わないようにしています。



TECHNIC 2

box-shadowでボタンに光沢をつける

box-shadowプロパティを使ってメニューボタンの内側に影をつけ、光沢感を表現しています。





ベースのHTMLを作成する

01

ここでは、サンプルをHTML5で作成していますが、IE8以下の古いブラウザはHTML5に対応していません。
このため、「html5.js」01-1という JavaScriptファイルを読み込むことでIE6～8にHTML5の要素を適用できるようにします。01-2のように入力し、「scripts」というフォルダにhtml5.jsを格納しておきます。
<!--[if lte IE 8]>～<![endif]-->で囲っているのは条件分岐コメントで、HTML5に対応していないIE8以下でhtml5.jsのファイルを読み込むように指示しています。

01-1 html5.js

```
http://html5shim.googlecode.com/svn/trunk/html5.js
```

01-2 html5.jsの読み込み（index.html）

```
<!--[if lte IE 8]>
<script src="./scripts/html5.js" type="text/javascript"></script>
<![endif]-->
```

02

次に、ブラウザによってCSSの各要素のデフォルト値が違うのを揃えるために、「html5reset-1.6.1.css」というファイルをダウンロードして読み込みます02-1。HTMLファイルでは02-2のように記述しました。

02-1 html5reset-1.6.1.css

```
http://code.google.com/p/html5resetcss/downloads/detail?name=html5reset-1.6.1.css&can=2&q=
```

02-2 CSSの読み込み（index.html）

```
<link rel="stylesheet" href="./styles/html5reset-1.6.1.css" type="text/css" />
<link rel="stylesheet" href="./styles/default.css" type="text/css" />
<link rel="stylesheet" href="./styles/content.css" type="text/css" />
<!--[if IE 8]>
<link rel="stylesheet" href="./styles/ie8.css" type="text/css" />
<![endif]-->
<!--[if lte IE 7]>
<link rel="stylesheet" href="./styles/ie6_7.css" type="text/css" />
<![endif]-->
```

MEMO

「content.css」はプルダウンメニューについての指定をまとめたCSSファイルです。ページ全体のレイアウトの大枠については「default.css」というファイルに記述しています。ここでは、CSSファイルはすべて「styles」フォルダに格納して、そこから読み込ませています。「ie6_7.css」と「ie8.css」については後述します。

03

bodyの中の基本的な構成は03-1のようになっています。ここまででページの基本的な構成は完成です。
aside要素のid="subContent"の場所に、今回のメインとなる横並びのプルダウンメニューが入ります。
次のステップから細かく説明していきます。

03-1 bodyの中の基本的な構成（index.html）

```
<div id="wrapper">
— 省略 —
<header id="header">
  — 省略 —
</header><!-- /#header -->
<div id="content">
  <section id="mainContent">
    — 省略 —
  </section><!-- /#mainConten -->
  <aside id="subContent">
   // ここが横並びのプルダウンメニューの
   場所です
  </aside><!-- /#subContent -->
</div><!-- /#content -->
</div><!-- /#wrapper -->
<footer id="footer">
— 省略 —
</footer><!-- /#footer -->
```

04

ナビゲーションのマークアップは、04-1のようにul要素を入れ子にしたシンプルなものになります。
正しくマークアップすることで、04-2のようにCSSがオフになった環境でもリストとして表示され機能を失わないようにしています。

04-2 CSSがオフでもリストとして表示される



04-1 ul要素を入れ子にする（index.html）

```
<nav>
 <ul id="menu">
  <li><a href="#">news</a></li>
  <li><a href="#">column</a>
   <ul>
    <li><a href="#">2012</a>
     <ul>
      <li><a href="#">02</a></li>
      <li><a href="#">01</a></li>
     </ul>
    </li>
    <li><a href="#">2011</a>
     <ul>
      <li><a href="#">12</a></li>
      <li><a href="#">11</a></li>
      <li><a href="#">10</a></li>
     </ul>
    </li>
   </ul>
  </li>
  ― 省略 ―
 </ul>
</nav>
```

MEMO

マークアップする際に注意すべき点はul要素を入れ子にするやり方です。01のようにli要素の外ではなく、02のように中に入れるようにします。

01 間違ったマークアップ

```
<ul>
 <li><a href="#">menu01</a></li>
  <ul>
   <li><a href="#">menu01-01</a></li>
   <li><a href="#">menu01-02</a></li>
   <li><a href="#">menu01-03</a></li>
  </ul>
</ul>
```

02 正しいマークアップ

```
<ul>
 <li><a href="#">menu01</a>
  <ul>
   <li><a href="#">menu01-01</a></li>
   <li><a href="#">menu01-02</a></li>
   <li><a href="#">menu01-03</a></li>
  </ul>
 </li>
</ul>
```

メニューの調整や装飾を行う

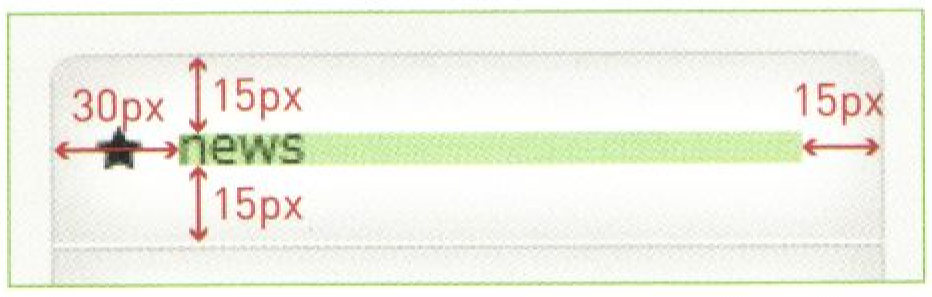
05

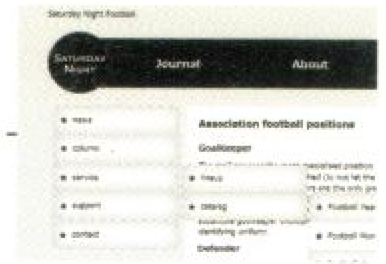
次に、CSSでナビゲーションの装飾をしていきます。メニューの横幅を200pxに設定し、li要素の中のa要素にpaddingを設定することでli要素全体をボタンとして反応するようにします05-1。paddingの設定を図で示すと、05-2のようになります。

05-1 li要素全体をボタンとして反応させる（content.css）

```
ul#menu {
 width: 200px;
}
ul#menu li {
 position: relative;
 font-size: 93%;
}
ul#menu li a {
 display: block;
 padding: 15px 15px 15px 30px;
 color: #000;
 text-decoration: none;
 ― 省略 ―
 background:
 #fff url(../images/common/nav_icon_01.
 gif) no-repeat 10px center;
 ― 省略 ―
}
```

05-2 paddingの設定





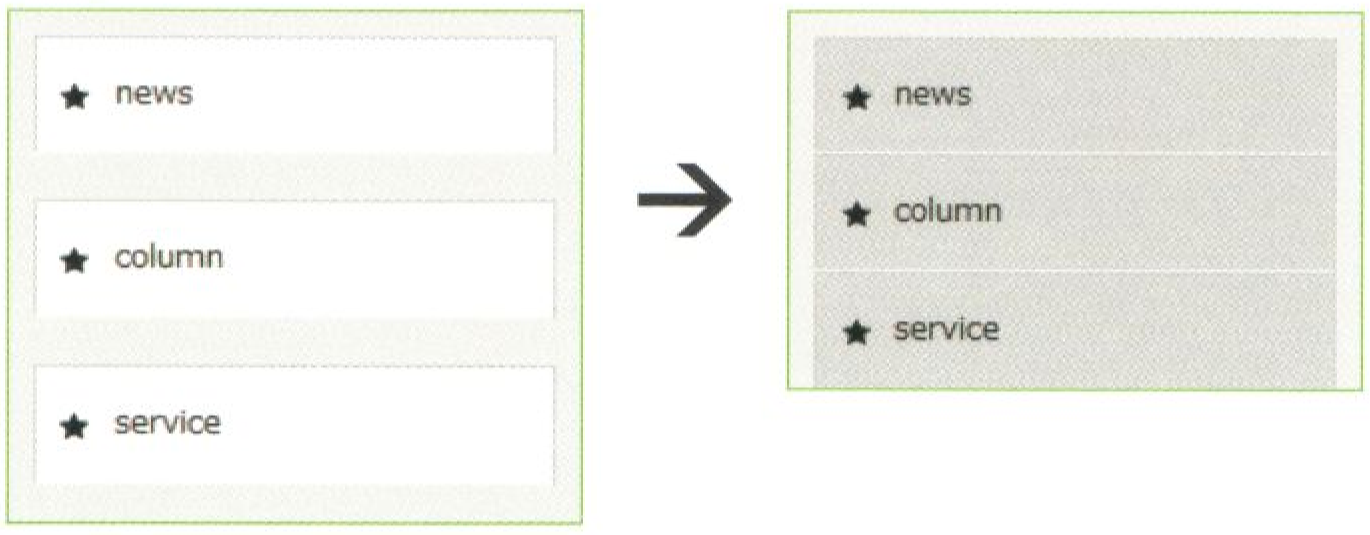
06

IE6～8で表示させるため、ie6_7.css（IE6～7用）、ie8.css（IE8用）を作成してCSSを上書き指定します 06-1 。li要素にdisplay: inline;を指定することでli要素の間の余白をなくします 06-2 。また、widthに全体の横幅（200px）から左右のpaddingで指定している幅（45px）を引いた155pxを指定します。こうすることで、li要素全体がボタンとして反応するようにします。

06-1 余白と幅の調整（ie6_7.css）

```
ul#menu li {
  display: inline;
}
ul#menu li a {
  width: 155px;
  ― 省略 ―
}
```

06-2 li要素の間の余白をなくす



07

次に、borderプロパティとbox-shadowプロパティを使って、ボタンに罫線や光沢を加えます。
さらに、:hoverでマウスがボタン上にあるときの指定をします。ここでは背景を白にして、box-shadowプロパティをなくしました 07-1 07-2 。

07-2 背景を白にする



07-1 ボタンの表現（content.css）

```
ul#menu li a {
 ― 省略 ―
  border-top: 1px solid #d2d2d2;
  border-bottom: 1px solid #f5f5f5;
  border-left: 1px solid #d2d2d2;
  border-right: 1px solid #d2d2d2;
  box-shadow: 0 0 25px #d2d2d2 inset;
 ― 省略 ―
}
ul#menu li a:hover {
  box-shadow: none;
  background: #fff url(../images/common/
  nav_icon_01.gif) no-repeat 10px center;
}
```

MEMO

box-shadowプロパティにinsetを指定すると、要素の内側に影が反映されます。これにより、 07-2 のような光沢のあるボタンを表現することが可能です。また、その他の値を変更するだけでいろいろな表現が行えるようになります。box-shadowの構文は 01 のようになっています。

01 box-shadowプロパティの構文

```
box-shadow: 水平方向 垂直方向 ぼかし距離 影の色 ;
```

08

box-shadowプロパティはIE6～8で対応していないため、そのまま何も指定しないと、真っ白なナビゲーションになります 08-1 。
IE6～8では影の表現はできなくなりますが、ボタンと認識しやすい最低限のロールオーバーはさせたいので、backgroundプロパティに色指定しておきます 08-2 08-3 。

08-1 IE6～8では真っ白なナビゲーションになる



08-2 背景色の指定（ie6_7.css）

```
ul#menu li a {
 ― 省略 ―
  background: #dcdcdc url(../images/common/nav_
  icon_01.gif) no-repeat 10px center;
}
```

08-3 背景色の指定（ie8.css）

```
ul#menu li a {
  background: #dcdcdc url(../images/common/nav_
  icon_01.gif) no-repeat 10px center;
}
```

09

次に、ナビゲーションを角丸の形状にします。擬似クラスを利用し、li要素の最初と最後の子要素に角丸をborder-radiusプロパティで適用します 09-1。

角丸を取り入れることでナビゲーションのひと固まりがよりわかりやすくなり、見た目の柔らかさも表現できます 09-2。

09-1 角丸の適用（content.css）

```
ul#menu li:last-child > a {
  border-bottom: 1px solid #d2d2d2;
}
ul#menu li:first-of-type > a {
  -webkit-border-radius: 7px 7px 0 0;
  -moz-border-radius: 7px 7px 0 0;
  border-radius: 7px 7px 0 0;
}
ul#menu li:last-of-type > a {
  -webkit-border-radius: 0 0 7px 7px;
  -moz-border-radius: 0 0 7px 7px;
  border-radius: 0 0 7px 7px;
}
```

09-2 ナビゲーションを角丸の形状に



MEMO

09-1で使用している:last-child擬似クラスは、指定した要素内で最後に書かれている子要素にスタイルを適用するものです。擬似クラスについてはINTRODUCTIONの18ページも参照ください。

MEMO

border-radiusプロパティはIE6～8で対応していませんが、角丸用のJavaScriptライブラリを読み込むなどの対策はとっていません。IE6～8での表示は01のようになります。

01 IE6～8では角丸が反映されない



メニューにプルダウン機能を実装する

10

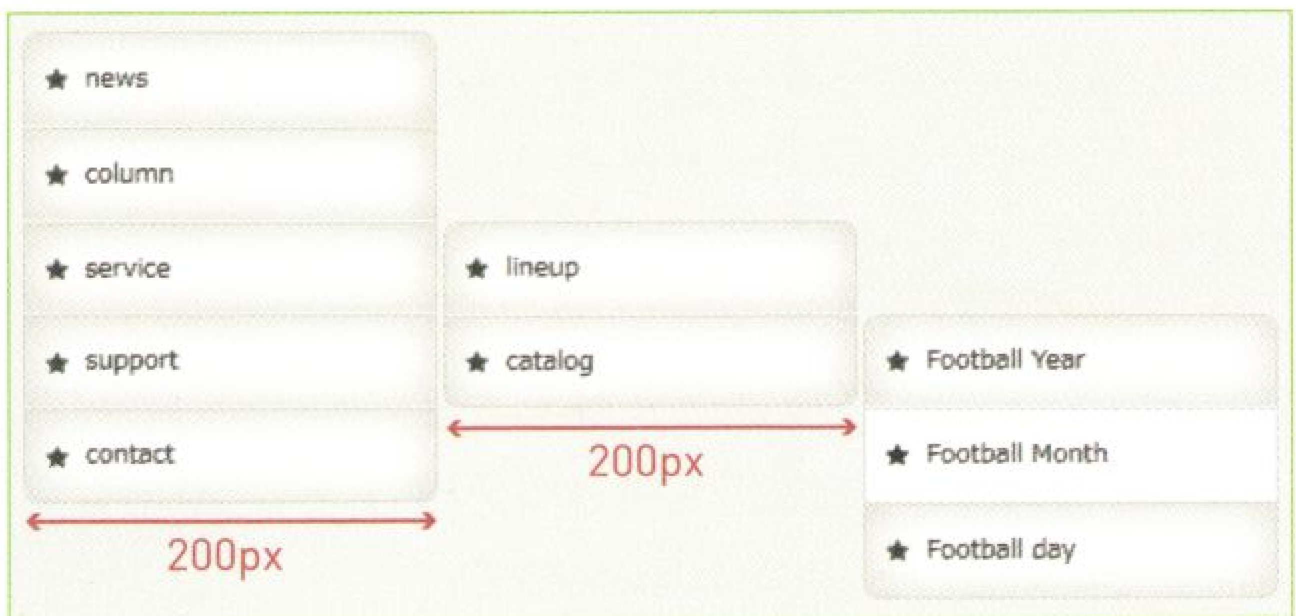
次に、メニューにプルダウン機能をつけていきます。li要素の中のul要素をdisplay: none;で非表示に、マウスがli要素上にあるときには、非表示にしているul要素をdisplay: block;で表示させているものです。

そして、li要素上にhoverしたときul要素を左から200pxずらした位置に表示させることでプルダウン機能を実現しています 10-1 10-2。

10-1 プルダウンの表現（content.css）

```
ul#menu li:hover > ul {
  display: block;
}
ul#menu ul {
  width: 200px;
  display: none;
  position: absolute;
  top: 0;
  left: 200px;
  padding-left: 3px;
  font-size: 108%;
}
```

10-2 ul要素を左から200pxずらすことでプルダウンメニューを実現





11

10にある「>」の子セレクタはIE6では非対応なので、ナビゲーションの基本的な構造を11-1のJavaScriptで補います（このjsファイルは筆者が書いたものですので、自由にお使いください）。ここでやっていることはCSSと同じで、li要素にhoverした際に非表示にしているul要素を表示しています。jQueryに依存する書き方なので、このJavaScriptの前にjQueryを読み込みます11-2。<!--[if IE 6]>〜<![endif]-->で囲ってブラウザが、IE6のときにのみ読み込ませる設定としました。

11-1 ie6.js

```
jQuery(function(){
  $('#menu li').hover(function(){
    $(this).children('ul').show();
  }, function(){
    $(this).children('ul').hide();
  });
});
```

11-2 ie6.jsの読み込み（index.html）

```
<!--[if IE 6]>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
<script src="./scripts/ie6.js" type="text/javascript"></script>
<![endif]-->
```

12

次に、メニューボタンにロールオーバーした際、ボタンの色がゆっくりと変化するのをCSSだけで表現します。ここでは、transition-durationプロパティで変化にかかる時間を指定しています12-1。これだけでJavaScriptを使ったような効果が得られます。
ただ、現状ではIEのすべてのバージョンでこの効果を得ることはできないため、注意が必要です。

12-1 変化の時間指定（content.css）

```
ul#menu li a {
  — 省略 —
  webkit-transition-duration: 0.3s;
  -moz-transition-duration: 0.3s;
  transition-duration: 0.3s;
}
```

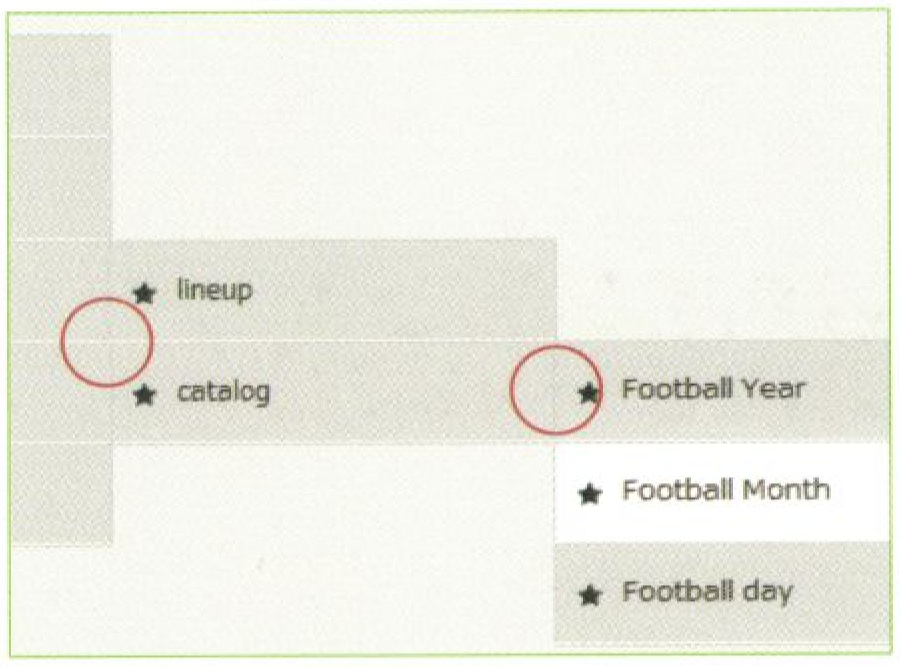
13

Firefox9とIE6で表示してみます。大きく違うのは角丸とボタンの装飾です。また、Firefoxではリストの間に余白がありますが、IEではありません13-1 13-2。IEではマウスポインタが余白上にあるとき、ul上にいないと判定されプルダウンがオフになります。ul上にいると判定させるためにIEではあえて余白をつけていません。

13-1 Firefox9では余白がある



13-2 IE6では余白がない



まとめ [CONCLUSION]

今回はできる限りIE6〜8でもプルダウンメニューとして機能を損なうことがないよう、簡単な対策をしてきました。ただ、13を見ていただいてもわかるように、Firefoxなどのモダンブラウザと呼ばれるものと同じ見た目ではありません。角丸を表現するborder-radiusプロパティや、変化するのにかかる時間を指定するtransition-durationプロパティなどは表現を豊かにしてくれますが、現状では未対応ブラウザへの対策を考えながら使用するどうか取捨選択する必要があります。

06 ロールオーバーによるナビゲーションの演出



ロールオーバーの手法は背景置換やJavaScriptライブラリを使うなど、さまざまなアプローチがあります。今回は画像を使わない、CSS3を利用する手法を紹介します。

制作・文／赤間公太郎（株式会社マジカルリミックス）

制作のポイント

- :hover擬似クラスをa要素以外に使用する
- opacityプロパティを使って透明度を調整する
- 複雑なグラデーションにはジェネレーターを利用する

対応ブラウザ

IE 6以上	Safari 4以上
Firefox 3以上	Chrome 12以上

TECHNIC 1

透明度を変える

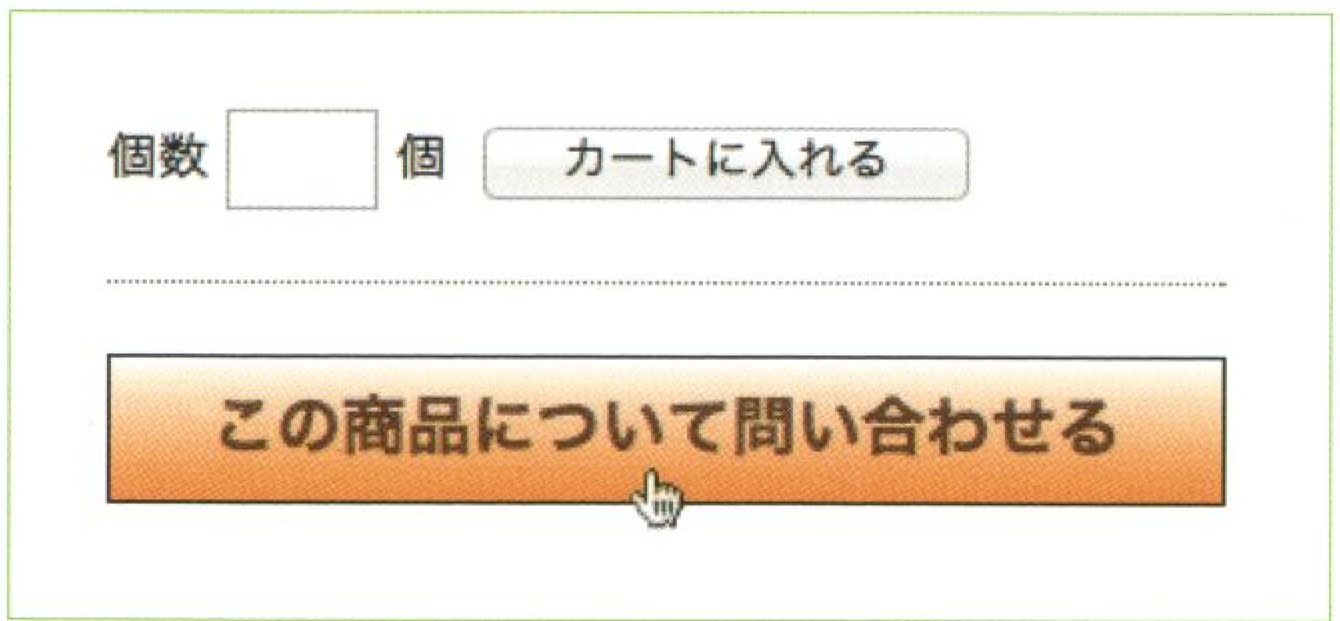
マウスが乗ったときに透明度を変えるだけでも、さりげないロールオーバー効果を演出することができます。



TECHNIC 2

グラデーションを設定する

グラデーションを設定し、ロールオーバー時の視覚効果を高めることができます。





ナビゲーションにロールオーバー効果を適用する

01

ナビゲーションにロールオーバー効果を適用しましょう。
まずは基本となるHTMLを用意します 01-1。

01-1 基本となるHTML

```
<div id="nav">
<ul>
<li><a href="#">お支払い方法</a></li>
<li><a href="#">配送について</a></li>
<li><a href="#">商品一覧</a></li>
<li><a href="#">カートの中身</a></li>
</ul>
</div>
```

02

CSSでナビゲーションらしいレイアウトに整えます 02-1 02-2。
このとき、a要素に対してdisplay: block;を指定し、さらにwidthとheightも指定します。これによりリンクの有効エリアが、指定した幅と高さの全面になります。
テキストの周りの余白部分も、リンクとしてクリックできるようにしましょう。

02-1 ナビゲーションをレイアウトするCSS

```
@charset "UTF-8";

body,h1,h2,h3,p,ul,ol,li,dl,dt,dd {
  margin:0;
  padding:0;
}
ol, ul { list-style:none; }
img {
  border: none;
  vertical-align:bottom
}
― 省略 ―
/*nav
------------------------------*/
#nav ul { height: 35px; }
#nav ul li { float: left; }
#nav ul li a {
  display: block;
  color: #e6e0da;
  background: #6a5441;
  width: 200px;
  height: 35px;
  line-height: 35px;
  text-align: center;
}
```

02-2 ナビゲーションらしいレイアウト



03

ロールオーバー効果で背景色を変更します。"マウスが乗ったとき"の指定はhover擬似クラスを使用します。

hover擬似クラスは、基本的にa要素に対して使用されますが、今回は極力シンプルなHTMLとCSSで構成するために、li要素に対してhoverを適用します 03-1。

03-1 li要素に対してhoverを適用

```
#nav ul li:hover {
}
```

MEMO

IE6では、a要素以外にhoverを設定しても無視されてしまいます。JavaScriptライブラリ「IE9.js」01を活用して、a要素以外にhoverを適用できるようにしましょう。
HTMLファイルのhead内に02を追記することで、IE8以下のバージョンでスクリプトが読み込まれます。
IE7.jsとIE8.jsもありますが、最新版はIE9.jsです。IE9.jsは、IE5.5からIE8までのHTML・CSS表示を、IE9のようによりW3C標準に近い形で再現できるライブラリです03。

01 ライブラリ「IE9.js」（サイト名：ie7-js）

http://code.google.com/p/ie7-js/

02 IE9.jsの読み込み

```
<!--[if lt IE 9]>
<script src="http://ie7-js.googlecode.com/svn/version/2.1(beta4)/IE9.js"></script>
<![endif]-->
```

03 IE9.jsの導入により利用できるセレクター覧

http://ie7-js.googlecode.com/svn/test/index.html

04

opacityプロパティを使ってロールオーバー効果を適用します 04-1 04-2。
opacityは要素を半透明にするプロパティです。
透明度を半分の50%にしたいときはopacity: 0.5;のように指定します。
IE向けには、独自プロパティのfilterを併用します。opacityプロパティとは数値の指定方法が違うので注意しましょう。

04-1 opacityおよびfilterプロパティで透明度を指定

```
#nav ul li:hover {
  opacity: 0.5;
  filter: alpha( opacity=50 );
}
```

04-2 マウスを乗せるとロールオーバー効果が現れる



TIPS

CSS3のtransitionプロパティを使用すると、ロールオーバー時にフワッと変化する時間効果を与えることができます。
IEは未対応ですが、01のようにベンダープレフィックスを利用すると、FirefoxやChromeなど主要なブラウザで有効です。

01 transitionプロパティをベンダープレフィックス付きで指定

```
#nav ul li:hover {
  opacity: 0.5;
  filter: alpha( opacity=50 );
  -moz-transition: 0.4s;        /* Firefox */
  -webkit-transition: 0.4s;     /* WebKit */
  -o-transition: 0.4s;          /* Opera */
  transition: 0.4s;             /* W3C */
}
```



*COLUMN

ベンダープレフィックス…-moz-、-webkit-などの接頭辞

CSS3のベンダープレフィックスは、ブラウザベンダーが草案段階の仕様を先行実装したり、独自の拡張機能を実装する場合に、それが拡張機能であることを明示するものです。
本書執筆時点ではベンダープレフィックスが必要ですが、今後ベンダープレフィックスを必要としなくなる場合に備えてベンダープレフィックスがアリの状態と、W3Cの標準プロパティを両方記述しておきましょう。
（ベンダープレフィックスについての解説は、15ページのINTRODUCTION「ブラウザとベンダープレフィックス」の項もご覧ください。）

リンクボタンにロールオーバー効果を適用する

05

次はリンクボタンに対してロールオーバー効果を適用していきます。
まず、問い合わせのリンクボタンを準備します。05-1のようにp要素とa要素で構成した、シンプルなリンクです。

05-1 問い合わせのリンクボタン

```
<p class="contact-btn"><a href="#">この商品について問い合わせる</a></p>
```

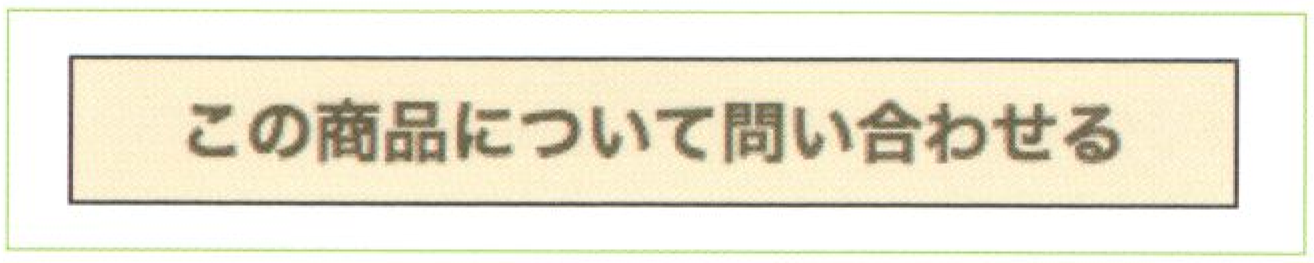
06

CSSでリンクボタンの見た目を整えます 06-1 06-2。
a要素にdisplay: block;を指定してブロック化し、リンクが有効なエリアを全面としました。
height: 40px;とline-height: 40px;のようにheightとline-heightの値を同じ数値にすると、垂直方向にセンタリングすることができます。
これにborder、backgroundなども指定します。

06-1 リンクボタンの見た目を整える

```
p.contact-btn a {
    display: block;
    background: #faecd4;
    border: 1px solid #000;
    width: 308px;
    height: 40px;
    line-height: 40px;
    text-align: center;
    font-size: 18px;
    font-weight: bold;
}
```

06-2 見た目を整えたリンクボタン



ロールオーバー時の効果をグラデーションにする

07

ロールオーバーしたときの効果をグラデーションにしてみましょう。
a要素にhover擬似クラスを使用してCSSを書いていきます。
グラデーションはlinear-gradient()関数を使用しますが、サポートされていないブラウザも視覚的にカバーするため、07-1のように、まずはロールオーバー時の基本的な背景色を指定しておきましょう。

07-1 ロールオーバー時の基本的な背景色を指定

```
p.contact-btn a:hover { background: #F9B445; }
```

08

linear-gradient()関数はCSS3の草案段階ですが、各ブラウザは拡張機能の形で先行実装しています。08-1のように、ベンダープレフィックスを付けましょう。
Chrome、Safariの古いバージョンでは、記述方法が若干異なるため、注意が必要です。

08-1 linear-gradient()関数にベンダープレフィックスを付ける

```
background: -moz-linear-gradient(); /* FF3.6+ */
background: -webkit-gradient(); /* Chrome,Safari4+ */
background: -webkit-linear-gradient(); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(); /* Opera 11.10+ */
background: -ms-linear-gradient(); /* IE10+ */
background: linear-gradient(); /* W3C */
```

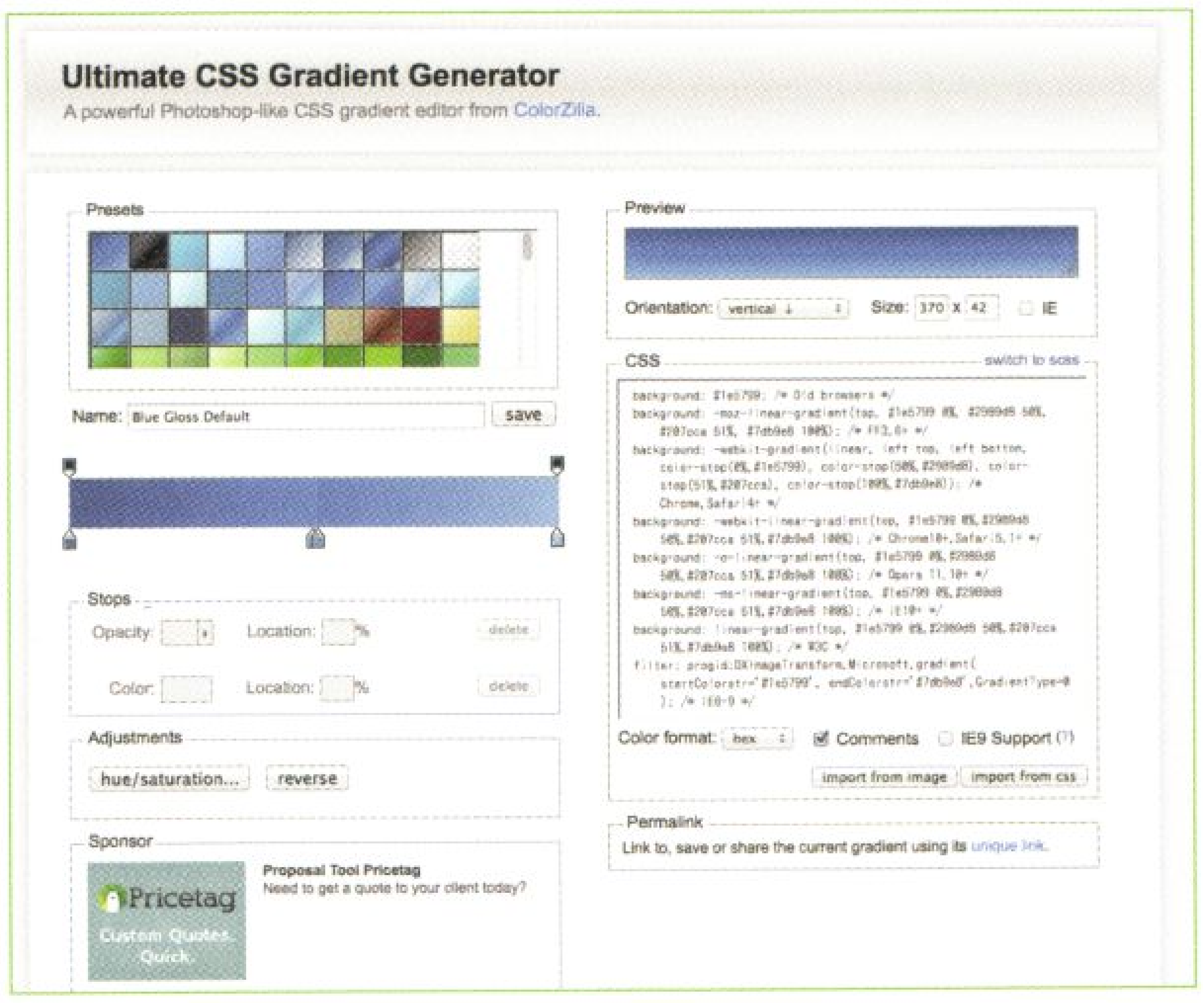
09

linear-gradient()関数は記述方法が若干複雑で、手書きによるコードは作業効率が悪いため、オンラインのジェネレーター（09-1など）を利用するケースが多いです。
特に複数の色を指定したり、適用範囲が細かいグラデーションを作成する場合に便利です。基本的にはマウス操作でグラデーションのプロパティを生成することができます09-2。
さらに、IE6にも対応するfilterプロパティも生成されるため、ほぼすべてのブラウザでグラデーションを表現することができます。

09-1 Ultimate CSS Gradient Generator

http://www.colorzilla.com/gradient-editor/

09-2 Ultimate CSS Gradient Generatorでグラデーションを作成

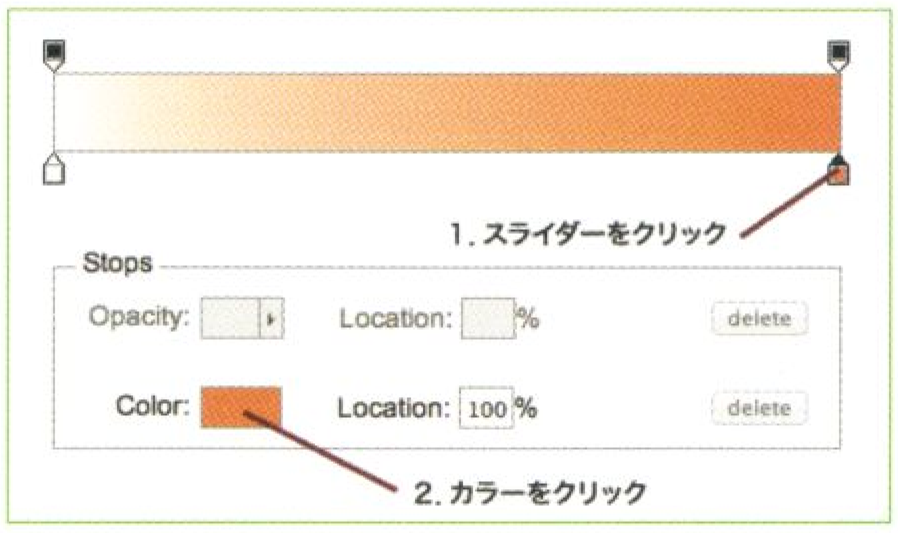




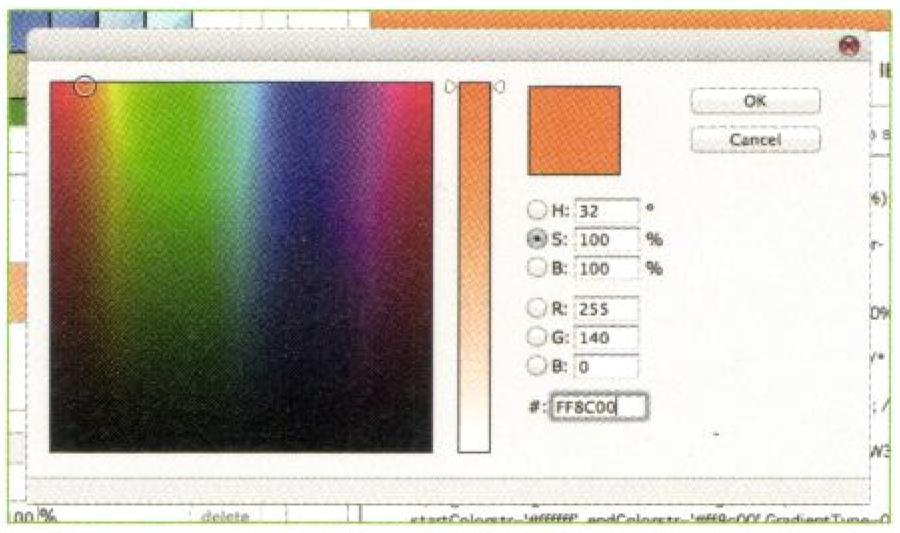
10

09の「Ultimate CSS Gradient Generator」を利用してグラデーションを作成してみましょう。
まず色を設定したいグラデーションスライダーをクリックします。次にすぐ下の「Stops」内にある「Color:」をクリックします 10-1。するとカラーピッカー 10-2 が現れるので、好きな色に設定します。
インターフェイスや設定方法はPhotoshopなどに近いので、グラフィックソフトで設定しているような操作性です。
今回は白からオレンジに変わるグラデーションを設定するため、開始位置のカラーを#FFFFFF、終了位置のカラーを#FF8C00としました。

10-1 グラデーションスライダー



10-2 カラーピッカー



11

ジェネレーターで生成されたコード 11-1 をコピーし、CSSファイルに貼り付けます。
「Ultimate CSS Gradient Generator」では、Firefox、Chrome、Safari、Opera、IE6～9、IE10、W3C標準を網羅したコードが生成されます 11-2。

11-1 ジェネレーターで生成されたコード

```
p.contact-btn a:hover {
  background: #F9B445; /* Old browsers */
  background: -moz-linear-gradient(top, #ffffff 0%, #ff8c00 100%); /* FF3.6+ */
  background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#ffffff),
  color-stop(100%,#ff8c00)); /* Chrome,Safari4+ */
  background: -webkit-linear-gradient(top, #ffffff 0%,#ff8c00 100%); /*
  Chrome10+,Safari5.1+ */
  background: -o-linear-gradient(top, #ffffff 0%,#ff8c00 100%); /* Opera 11.10+ */
  background: -ms-linear-gradient(top, #ffffff 0%,#ff8c00 100%); /* IE10+ */
  background: linear-gradient(top, #ffffff 0%,#ff8c00 100%); /* W3C */
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#ffffff', endColorstr='
  #ff8c00',GradientType=0 ); /* IE6-9 */
}
```

11-2 グラデーションが表現されたリンクボタン



まとめ [CONCLUSION]

マウスを乗せたとき、透明度を少し変化させるだけでも視覚効果があるのでロールオーバー効果として活用できます。透明になったとき、背景の色で「明るく」、「暗く」などが工夫できますので、親要素の背景色をイロイロ指定してみるのもよいでしょう。

グラデーションのコードを作成する際には、オンラインジェネレーターを活用すると便利です。
手打ちではとうてい書き切れないような、細かい設定とクロスブラウザ対応のコードがカンタンに生成可能です。

07 ユーザビリティの高い iPhone風UIフォーム



スマートフォンサイトの入力フォームは、ブラウザのデフォルトで用意されているものそのままでは、使いやすさに難があります。CSS3を使ってiPhoneのアプリのようなフォームを実装し、ユーザーの使い勝手を向上させましょう。

制作・文／執行大介

制作のポイント

- 「-webkit-appearance:none;」によるフォームのノーマライズ（初期化）
- 擬似要素セレクタ:checkedとtransitionプロパティによる表示の切り替え
- アイコンなどは無理にCSS3による表現を使わず、画像を用いる

対応ブラウザ

iOS	3.0 以上
Android	2.1 以上

※Androidは一部対応していない機種があります。

TECHNIC 1

iPhoneライクなUIを作成する

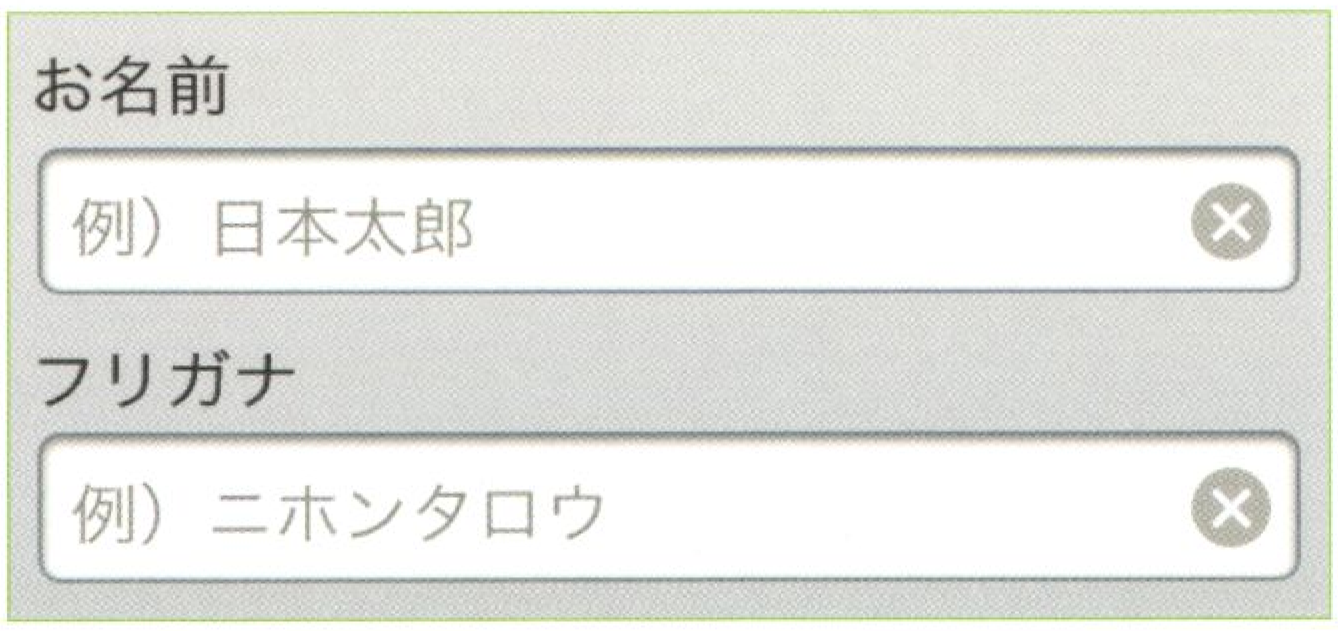
フォームの各要素はブラウザの初期の表示を使わず、ユーザーが使い慣れたiPhoneライクなUIを作成します。



TECHNIC 2

入力内容のクリアボタンを用意する

テキストフィールドの入力内容をクリアできるボタン⊗を用意することで、ユーザーの使いやすさを向上させます。





ベースとなるHTMLの記述

01

最初にフォームのHTMLにとりかかります。01-1のコードのformタグ内に01-2のように記述していきます。まずは名前などのテキストフィールドを作り、要素に「placeholder="…"」を入力することでテキストフィールドに入力例などを記述しましょう。また、inputタグの後にanchorタグ（以下aタグ）を記述します。これが入力内容をクリアするためのボタンになります。

01-1 ベースとなるHTML（index.html）

```
<!DOCTYPE HTML>
<html lang="ja">
<head>
<meta charset="UTF-8">
<title>スマートフォンサイトでよく使う便利なUIフォーム</title>
<meta name="keywords" content="">
<meta name="description" content="">

<link rel="index" href="/">

<meta name="format-detection" content="telephone=no">
<meta name="viewport" content="width=device-width, user-
scalable=yes, initial-scale=1, maximum-scale=1"/>
<meta name="apple-mobile-web-app-capable" content="yes">

<link rel="stylesheet" href="css/common.css">
<link rel="stylesheet" href="css/form.css">
</head>

<body>
<header>
<h1 class="globalHeaderLogo"><img src="images/no1_logo.png"
alt="Studio Panda Space"></h1>
</header>

<article class="mainArticle">

<section class="mainHead">
<h1>■申し込みフォーム</h1>
<p>申し込みはこちらからおねがいします。</p>
</section>

<section class="mainSec">
<form action="#">
// ここに01-2の内容を記述
</form>

<!-- / .mainSec --></section>
<!-- / .mainArticle --></article>

<footer class="globalFooter">
<p class="copyringht">Studio Panda Space</p>
<!-- / .globalFooter --></footer>
</body>
</html>
```

01-2 index.htmlのformタグ内に記述する内容

```
<h2 class="checkTitle">お名前</h2>
<p class="formText"><input class="name" type="text" placeholder="例）日本太郎" value=""><a class="inputTextCL" href="#">×</a></p>

<h2 class="checkTitle">フリガナ</h2>
<p class="formText"><label><input class="furi" type="text" placeholder="例）ニホンタロウ" value=""><a class="inputTextCL" href="#">×</a>
</label></p>

<h2 class="checkTitle">メールアドレス</h2>
<p class="formText"><label><input class="address" type="email" placeholder="例）taro@panda-space.com" value=""><a class="inputTextCL"
href="#">×</a></label></p>
```

TIPS

「プレースホルダー」とは、テキスト入力フォームに最初から表示されているテキストのことです。HTML5から導入された要素名「placeholder」に値を入れることで、入力例などを表示でき、テキストを入力する際には自動で非表示になります。

MEMO

このサンプルでは、2つのCSSファイルを読み込んでいます。「common.css」は全般的なスタイルの設定、「form.css」は個別要素に対するスタイルの設定を行っていますが、本書では「form.css」の内容だけを解説します。

テキストフィールドを実装する

02

次にCSSを別ファイルに記述します。「common.css」は一般的なスマートフォンサイトの設定ですが、iPhone4以降のRetinaディスプレイ対応のため「img」要素に「zoom: .5;」を指定しています。次は「form.css」02-1でフォームの装飾を行います。

まずはテキストフィールドを「-webkit-appearance:none;」で初期化します。さらに「display:block;」でブロック要素にし、パディングと幅を指定してテキストフィールドの大きさを決めます。また、iOS／Android端末を横に傾けて画面の幅が変化した場合も、左右のpaddingを一定にするため、「box-sizing: border-box;」を指定しました。

02-1 テキストフィールドのスタイル①（form.css）

```
input[type="text"], input[type="email"]{
  /*テキストフィールドの初期化*/
  -webkit-appearance:none;

  /*大きさを決める*/
  display:block;
  padding:7px 4px;
  width:100%;

  /*ボックスにpaddingを含めないようにする*/
  -webkit-box-sizing: border-box;
  box-sizing: border-box;

  ― 省略（03と04で解説） ―
}
```

03

次に、フォントサイズなどを指定し、「box-shadow: inset …」で内側の影をつけます03-1。影の数値指定は通常の「box-shadow」と同じですが、その前に「inset」をつけることで影が内向きになります。

ATTENTION!

Androidでは、機種により「box-shadow」の「inset」に対応していないものもあります。

03-1 テキストフィールドのスタイル②（form.css）

```
input[type="text"], input[type="email"]{
  ― 省略（02で解説） ―

  /*フォントサイズと行の高さを指定*/
  font-size:16px;
  line-height:1.2em;

  /*内側の影をつける*/
  -webkit-box-shadow: inset 1px 1px 3px rgba(0, 0, 0, 0.4);
  box-shadow: inset 1px 1px 3px rgba(0, 0, 0, 0.4);

  ― 省略（04で解説） ―
}
```

04

さらに「border-radius」で角丸を指定し、外側にボーダーをつけます04-1。今回は四辺がすべて同じ値でよいので、指定する数値は1つで済みます。

04-1 テキストフィールドのスタイル③（form.css）

```
input[type="text"], input[type="email"]{
  ― 省略（02と03で解説） ―

  /*角丸を5pxで指定する*/
  -webkit-border-radius: 5px;
  border-radius: 5px;

  /*外側のボーダーをつける*/
  border:#5d738e 2px solid;
}
```



05

プレースホルダーのCSSは「input::-webkit-input-placeholder」というセレクタで指定可能です 05-1。入力するときの文字より色を薄くすることで、初期入力されている文字であることをわかりやすくしておきましょう 05-2。

05-1 プレースホルダーのスタイル（form.css）

```
/*プレースホルダーの色を指定する*/
input::-webkit-input-placeholder{
  color:#808080;
}
```

05-2 テキストフィールドのプレースホルダー



06

テキストフィールド部分の最後に、入力された文字のクリアボタンを作ります。「×」のアイコンはデバイスのテキストとCSSで作ることも可能ですが、Androidでは角丸が綺麗ではなく、フォントが変わってしまうため、ここではaタグの背景画像で指定しました。「position:absolute;」と「right:7px;」で右側に配置して、「display:inline-block;」で要素の大きさを確保しています。さらにこのHTML自体がiPhone4以降のRetinaディスプレイにも対応するものになっており、全体が半分の大きさになっているので、CSSの背景画像もで半分の大きさ（ここでは「background-size:128px 60px;」）で表示させます 06-1。

入力された文字をクリアする仕組みは後述するJavaScriptで実装します。

06-1 クリアボタンのスタイル（form.css）

```
.formText a{
  /*絶対配置で右側に寄せる*/
  position:absolute;
  top:9px;
  right:7px;

  /*要素の大きさを指定する*/
  display:inline-block;
  width:20px;
  height:20px;
  text-indent:-9999px;

  /*アイコンの背景画像を指定する*/
  background:url(../images/sp_icons.png)
  0px 0px no-repeat #fff;

  /*iPhone4対応のため半分の大きさで表示*/
  -webkit-background-size:128px 60px;
  background-size:128px 60px;
}
```

MEMO

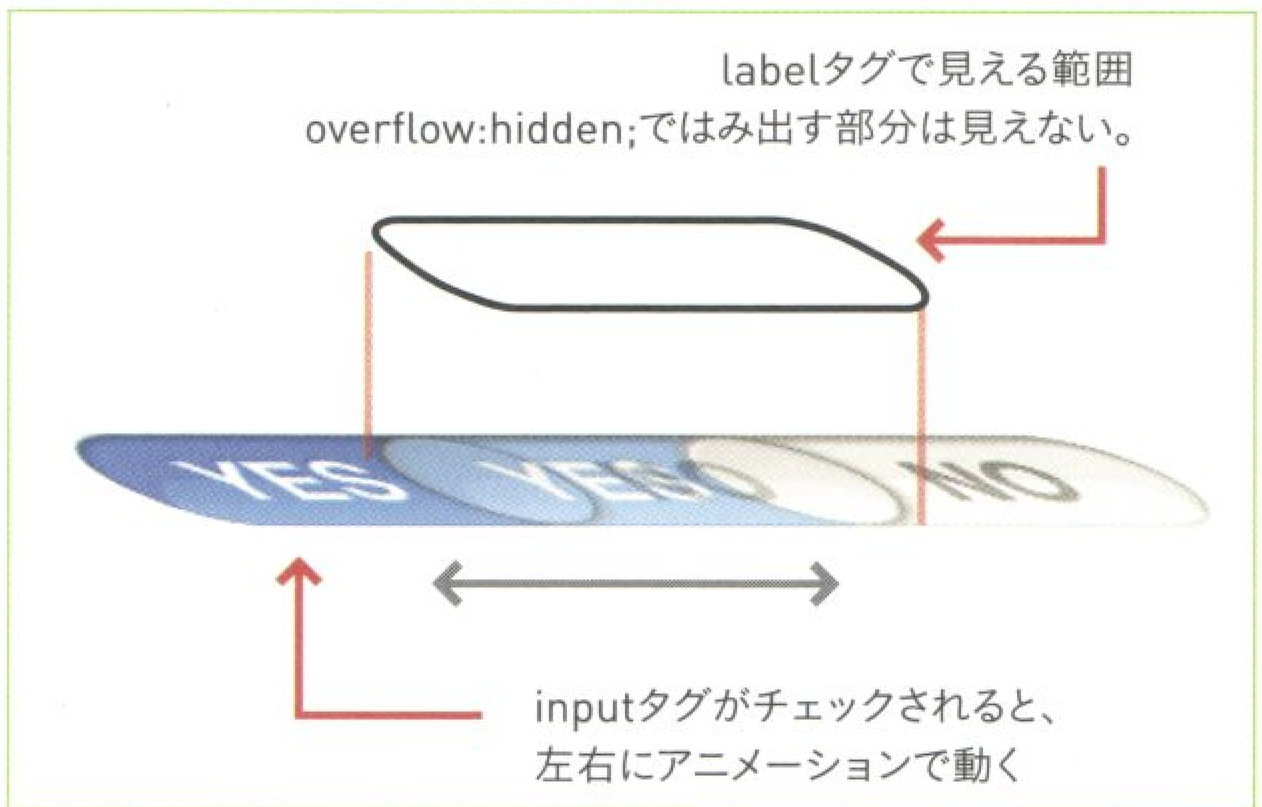
PCで鮮明に見えている画像でも、高精細なRetinaディスプレイではぼやけて見えてしまうことがあります。その対策として、大きなサイズの画像を用意して50%のサイズで表示させるという方法がとられます。ここでは256px×120pxの画像を128px×60pxで表示させています。

チェックボックスを実装

07

次にチェックボックスの実装を行います。iPhoneの設定画面では、ONとOFFをタップするごとに左右にアニメーションで切り替えるUIが採用されています。このUIによりユーザーはタップしやすく、ONかOFFか非常にわかりやすくなっています。今回はチェックボックスを、このUIで実装してみましょう 07-1。

07-1 iPhoneライクなUIを実現する仕組み



08

HTMLは 08-1 のようにpタグの中にlabelタグを記述し、チェックボックスとなるinputタグをさらに、その中（子要素）に書くのがポイントです。

08-1 チェックボックス（index.html）

```
<h2 class="checkTitle">メールマガジンの購読</h2>
<p class="formCheck"><label><input type="checkbox" name="mailMag" value=""></
label></p>
```

09

pタグを絶対配置で右に寄せ、labelタグを実際にタップするボックスにします。角丸とボーダーをつけ、「overflow: hidden;」で内包する「input」要素がはみ出る部分を隠すようにします 09-1。

09-1 チェックボックスのスタイル ①（form.css）

```
.formCheck{
  /*絶対配置で右側に寄せる*/
  position:absolute;
  top:9px;
  right:12px;
  display:block;
}
.formCheck>label{
  position:relative;
  display:block;
  width:78px;
  height:28px;

  /*角丸を高さの半分で指定*/
  -webkit-border-radius: 14px;
  border-radius: 14px;
  border:#7e7e7e 1px solid;

  /*内包する要素がはみ出ないようにする*/
  overflow:hidden;
}
```

10

チェックボックスもテキストフィールド同様「-webkit-appearance:none;」で初期化し、背景画像も半分の大きさで指定します。「transition」で「left」の位置がアニメーションで動くように設定して、セレクタ「input[type="checkbox"]:checked」で変化後の「left」の値を指定します。チェックボックスをタップするとチェックボックスが「:checked」の状態になり、画像が右にアニメーションするという仕組みです 10-1。

10-1 チェックボックスのスタイル ②（form.css）

```
input[type="checkbox"]{
  /*チェックボックスの初期化*/
  -webkit-appearance:none;

  /*絶対配置で設定しておく*/
  position:absolute;
  top:-1px;
  left:-1px;
  margin:0px;

  width: 78px;
  height:28px;

  /*背景画像を設定*/
  background:url(../images/sp_icons.png) -51px -30px no-repeat
  #eee;

  /*iPhone4対応のため半分の大きさで表示*/
  -webkit-background-size:128px 60px;
  background-size:128px 60px;

  -webkit-border-radius: 14px;
  border-radius: 14px;

  border:#7e7e7e 1px solid;

  /*トランジションで位置が動くようにする*/
  -webkit-transition-property: left;
  -webkit-transition-duration: 0.2s;
  -webkit-transition-timing-function: ease-in-out;
}
input[type="checkbox"]:checked{
  left:-30px;
}
```



ラジオボタンを実装する

11

次にラジオボタンを実装します。HTMLではリスト（li）タグ内に3つのラジオボタンを記述しています。さらにチェックボックスとは異なり、「label」タグを「input」タグのあとに兄弟要素とするのがラジオボタンのポイントです11-1。

11-1 ラジオボタン（index.html）

```
<h2 class="checkTitle">メールマガジンの形式</h2>
<ul class="radioSet">
 <li><input type="radio" name="mailType" id="no1" value="テキスト"><label for="no1">テキスト</label></li>
 <li><input type="radio" name="mailType" id="no2" value="HTML"><label for="no2">HTML</label></li>
 <li><input type="radio" name="mailType" id="no3" value="携帯"><label for="no3">携帯</label></li>
</ul>
```

12

今回リストタグを横に並べるのに、親要素のulタグに「display:box;」を使いました。これは「float」を使わずに内包する要素を横並びにできるプロパティです。内包するリストタグが3分割になるように「box-flex:1;」を指定します12-1。

ATTENTION!

2012年1月現在、「display:box;」は仕様が変更になり「display:flexbox;」という名前に変わっていますが、まだどのスマートフォンも実装できていないため両方を併記しておきます。「box-flex」も今後変更される可能性が高いと思いますので、心配な場合は「width」で幅を指定しておきましょう。

12-1 ラジオボタンのスタイル ①（form.css）

```
.radioSet{
  /*内包する要素を横並びにする*/
  display:-webkit-box;
  display:-webkit-flexbox;
  display:flexbox;

  margin:6px 0px 0px;
  width:100%;

  -webkit-border-radius: 5px;
 border-radius: 5px;

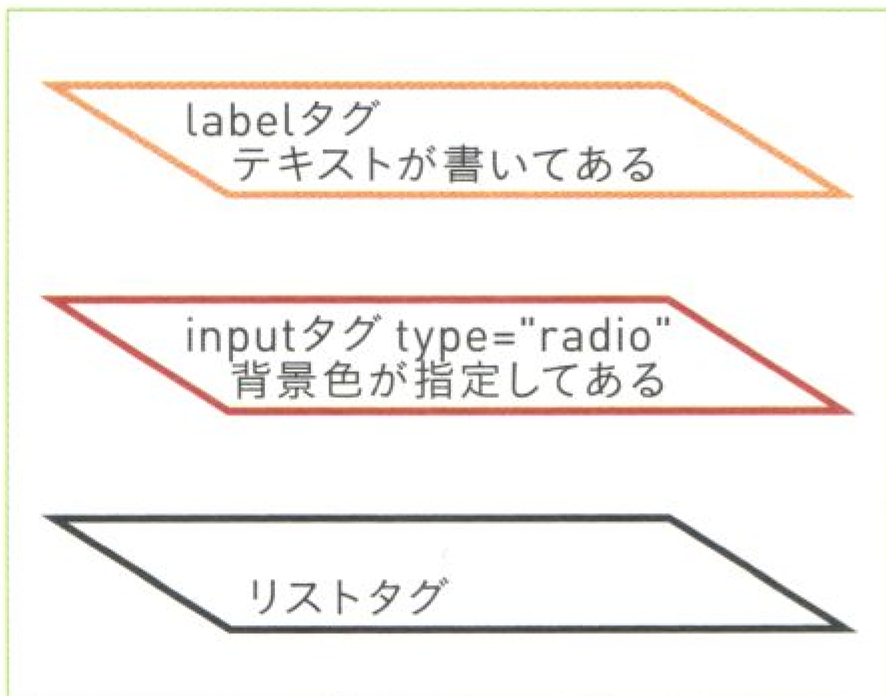
  border:#54677d 1px solid;

  overflow: hidden;
}
.radioSet li{
  /*幅を3分割想定で指定する*/
  -webkit-box-flex:1;
  box-flex:1;
  height:28px;
  border-right:#54677d 1px solid;
}
.radioSet li:last-child{
  /*最後のリストタグは左のボーダーをなくす*/
  border-right:none;
}
```

13

リストタグの中は「-webkit-appearance:none;」で初期化したラジオボタンと、グラデーションの背景を指定したlabelタグを絶対配置で重ねています13-1。スマートフォンのブラウザはlabelタグをタップしてもinputタグに反映されないものが多いですが、このように重ねておけば問題はありません。また、セレクタの兄弟要素指定「input[type="radio"]+label」と擬似要素「:checked」を用いることで、チェックされたラジオボタンの背景と文字を変えるようになっています13-2。

13-1 タグを重ねている



13-2 ラジオボタンのスタイル②（form.css）

```
.radioSet li input[type="radio"]{
  /*ラジオボタンの初期化*/
  -webkit-appearance:none;

  /*絶対配置で設定しておく*/
  position:absolute;

  display:inline-block;
  margin:0px;
  padding:0px;
  width:100%;
  height:28px;
  border:none;
  z-index:10;
}
.radioSet li input[type="radio"]+label{
  /*絶対配置で設定しておく*/
  position:absolute;

  display:inline-block;
  width:100%;
  height:28px;

  font-size:13px;
  line-height:28px;
  text-align:center;
  color:#3f5c84;

  text-shadow:0px 1px 1px rgba(255,255,255,.4);

  /*未選択時の背景グラデーション*/
  background-image: -webkit-gradient(linear, left top, left bottom,
  color-stop(0%, #d4dbdf), color-stop(100%, #b4bec6));
  background-image: -webkit-linear-gradient(-90deg, #d4dbdf 0%,
  #b4bec6 100%);
  background-image: linear-gradient(-90deg, #d4dbdf 0%, #b4bec6
  100%);

  z-index:1;
}
.radioSet li:first-child input[type="radio"]+label{
  -webkit-border-radius: 4px 0px 0px 4px;
  border-radius: 4px 0px 0px 4px;
}
.radioSet li:last-child input[type="radio"]+label{
  -webkit-border-radius: 0px 4px 4px 0px;
  border-radius: 0px 4px 4px 0px;
}

/*ラジオボタンがチェックされたときの
/ その兄弟要素のlabelのセレクタ指定*/
.radioSet li input[type="radio"]:checked+label{
  color:#FFF;
  /*選択時の背景グラデーション*/
  background-image: -webkit-gradient(linear, left top, left bottom,
  color-stop(0%, #8c9aae), color-stop(100%, #647792));
  background-image: -webkit-linear-gradient(-90deg, #8c9aae 0%,
  #647792 100%);
  background-image: linear-gradient(-90deg, #8c9aae 0%, #647792
  100%);
}
```

送信ボタンを実装する

14

次に送信ボタンを実装します。こちらはHTMLは通常のものとまったく変わりません 14-1。CSSではsubmitボタンを「-webkit-appearance:none;」で初期化し、「display:table-cell;」を指定することでテキストを縦中央配置である「vertical-align:middle;」を指定しています 14-2。背景のグラデーションは、iPhoneライクなボタンにするために4色のグラデーションにしました。

14-1 送信ボタン（index.html）

```
<input type="submit" value="送信">
```



14-2 送信ボタンのスタイル（form.css）

```
input[type="submit"]{
  /*ラジオボタンの初期化*/
  -webkit-appearance:none;
  /*テキストを縦中央に配置する用意*/
  display:table-cell;

  width:100%;
  height:46px;

  font-size:20px;
  /*テキストを縦中央に配置する*/
  vertical-align:middle;

  /*複数指定のグラデーションで
  / iPhoneライクなボタンを表現する*/
  background-image: -webkit-gradient(linear, left top, left bottom,
  color-stop(0%, #fbfbfb), color-stop(49%, #d2d3d5), color-stop(50%,
  #c0c1c4), color-stop(100%, #a9aaad));
  background-image: -moz-linear-gradient(0deg, #fdfdfd 0%,
  #d2d3d5 49%, #c0c1c4 50%, #a9aaad 100%);
  background-image: linear-gradient(0deg, #fdfdfd 0%, #d2d3d5
  49%, #c0c1c4 50%, #a9aaad 100%);

  -webkit-border-radius: 14px;
  border-radius: 14px;

  border:#464951 3px solid;
}
```

テキストフィールドをクリアするJavaScriptの実装

15

最後に、テキストフィールドの入力をクリアするJavaScriptを実装します 15-1。aタグである「×」をクリックすると「.prev()」でinputタグの内容を操作でき、「.val('')」で中身をクリアしています。最後に「return false;」を記述することで、aタグによるページ先頭へのリンクをキャンセルしている仕組みです。

MEMO

今回のサンプルでは、JavaScript ライブラリである「jQuery」01を使っているため、「<script src="js/jquery.js?ver=1.7"></script>」でjQueryを読み込んでいます。

01 jQuery

http://jquery.com/

15-1 テキストフィールドをクリアするJavaScript（index.html）

```
<script src="js/jquery.js?ver=1.7"></script>

<script type="text/javascript">
// DOMの読み込み完了を待つ
$(function(){

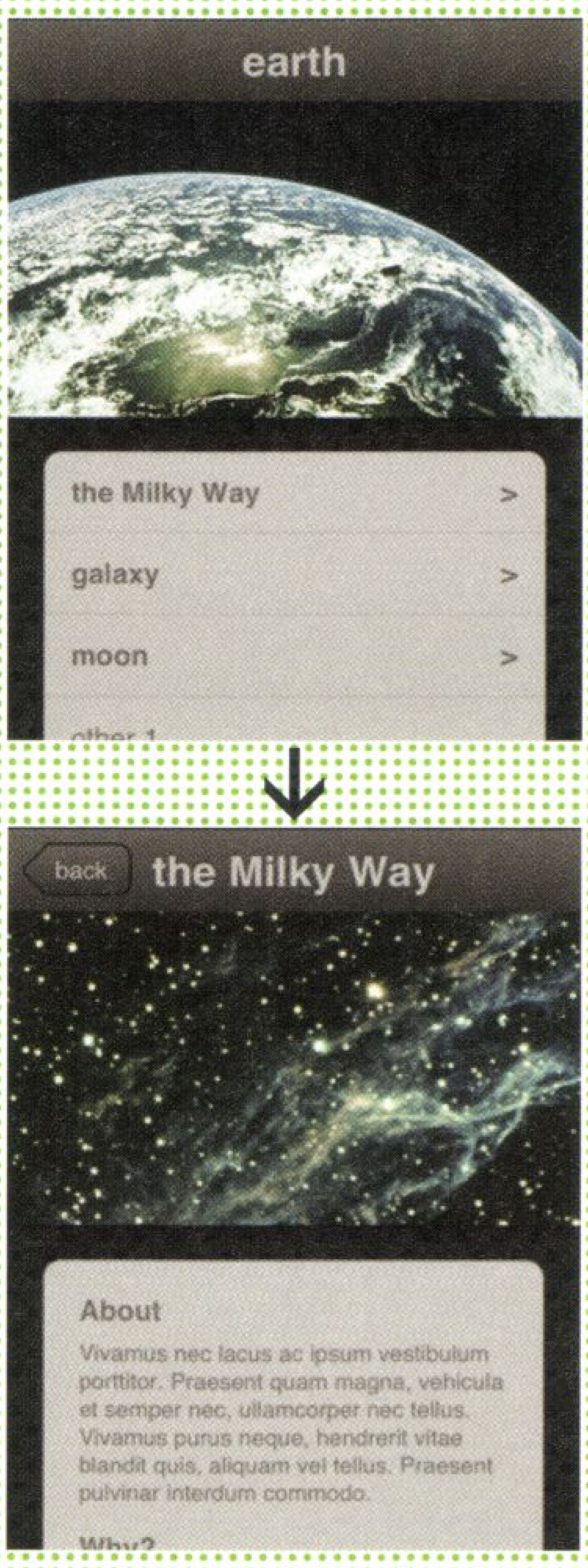
  // aタグのclass [inputTextCL]にclickイベントをつける
  $('.inputTextCL').bind('click',function(){
    // aタグの1つ前の要素の
    //inputタグの[value]の中身を空（''）にする
    $(this).prev().val('');

    // aタグのデフォルトの機能をなくす
    return false;
  });
});
</script>
```

まとめ [CONCLUSION]

フォームはUIの良し悪しでユーザーの反応が如実に変わるコンテンツです。スマートフォン対応のポイントは「タップしやすい大きめの要素」と「使い慣れたインターフェイス」の2つに尽きます。CSS3で実装する場合、アイコンなどの細かいところを表現しようと無理をしてしまいますが、アイコンなどはあえて画像を使うことを選択して、そのぶん動きなどに注力をしたほうが、いろいろなデバイスの対応もしやすくなり一貫性のあるUIを実現できます。

08 ページをスライドさせるリスト型ナビゲーション



スマートフォン向けサイトの定番でもあるリスト型のナビゲーション。ここではコンテンツをスライドさせて表示する仕組みを紹介します。

制作・文／山本圭助
（株式会社ラナデザインアソシエイツ）

制作のポイント

- コンテンツを左右に移動させて内容を切り替える
- JavaScriptとtransitionプロパティでスライドを実装する
- ヘッダーの内容や、メインビジュアルも切り替える

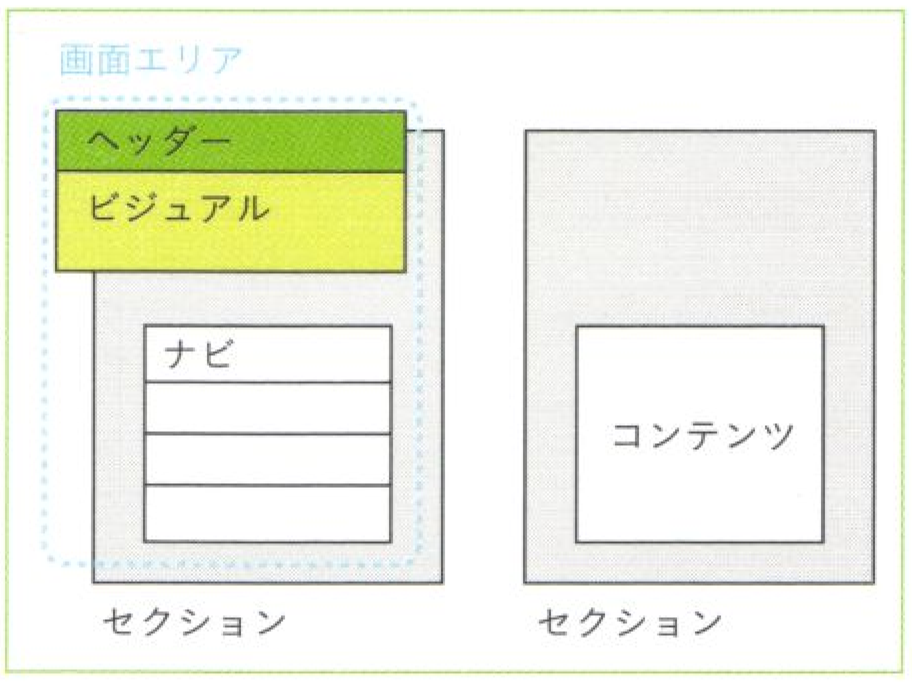
対応ブラウザ

iOS	4.0 以上
Android	1.6 以上

TECHNIC 1

セクションでコンテンツを囲って移動

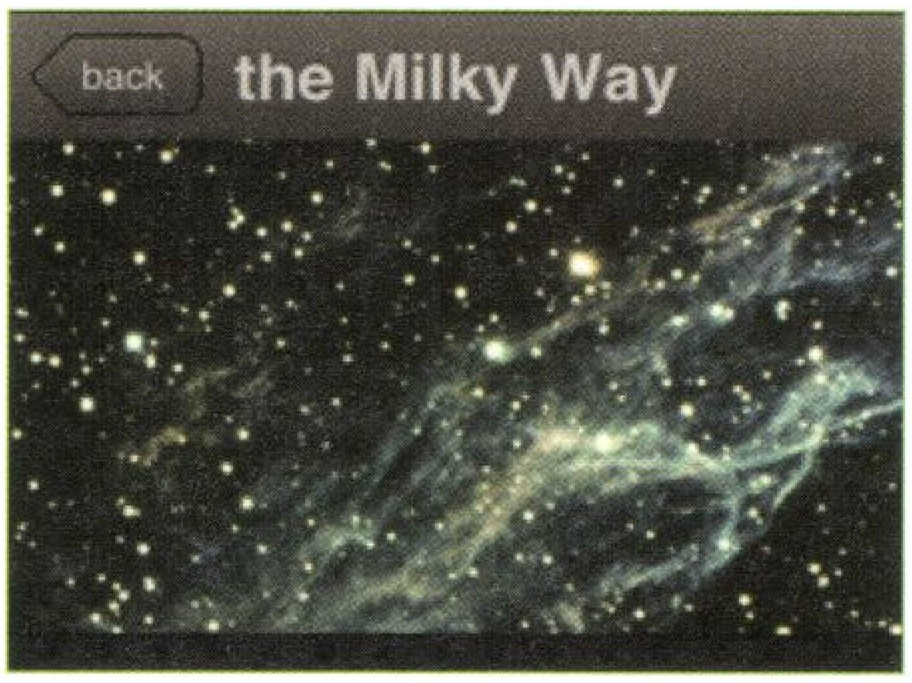
横並びに配置した各コンテンツを透明なセクションで囲い、leftプロパティの値を変化させることで、ページ全体を移動させています。

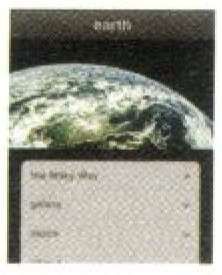


TECHNIC 2

ヘッダーの内容もページごとに変更

各ページのヘッダーの見出しや、「戻る」ボタンの表示／非表示も、CSSを使って実現できます。



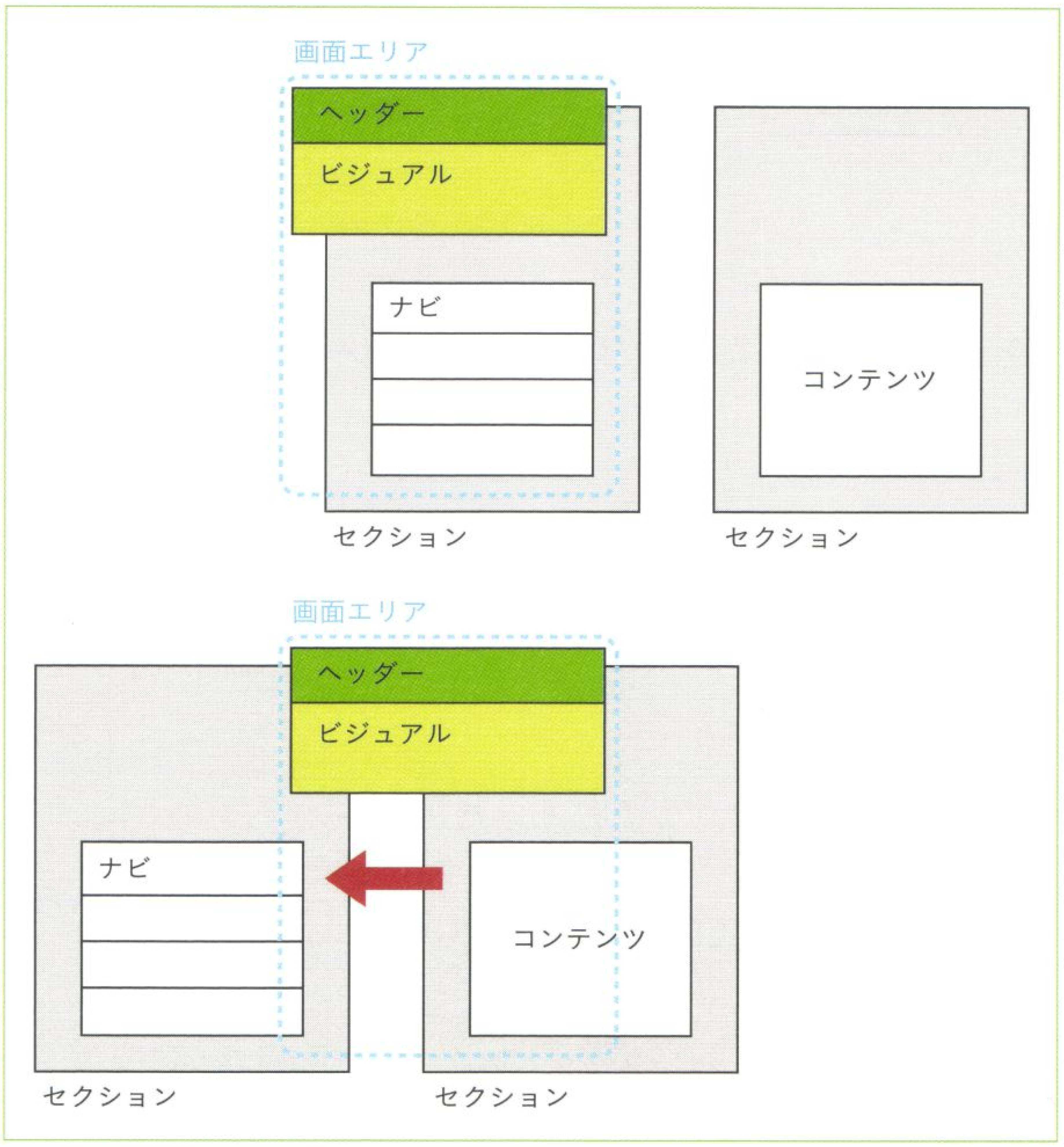


構成するページの作成

01

これから作成するページの全体構成は、01-1のようになります。
各コンテンツを透明なセクションで囲い、それを移動させることでページ全体を移動させます。
ページとなるセクションと、目次となるリストは01-2のように配置します。

01-1 Webページの全体構成



01-2 セクションとリストの配置

```
<section id="page-0">
  // 目次
  <nav>
    <ul>
      <li><a href="#page-1">1ページ目へ</a></li>
      <li><a href="#page-2">2ページ目へ</a></li>
      <li><a href="#page-3">3ページ目へ</a></li>
    </ul>
  </nav>
</section>

<section id="page-1">
  <div class="content">
    // 1ページ目
  </div>
</section>

<section id="page-2">
  <div class="content">
    // 2ページ目
  </div>
</section>

<section id="page-3">
  <div class="content">
    // 3ページ目
  </div>
</section>
```

02

各ページ（section要素）をくくるラッパー要素（#wrapper）も用意しておきます 02-1 02-2。
このラッパー要素のoverflowプロパティをhiddenに設定することにより、スマートフォンで横幅が画面にぴったりフィットするようになります。

02-1 ラッパー要素の準備（HTML）

```
// html
<body>
  // ラッパー要素
  <div id="wrapper">
    // 目次
    <section id="page-0"> ... </section>
    // 1ページ目
    <section id="page-1"> ...  </section>
    // 2ページ目
    <section id="page-2"> ... </section>
    // 3ページ目
    <section id="page-3"> ... </section>
  </div>
</body>
```

02-2 ラッパー要素の準備（CSS）

```
// css
#wrapper {
  position: relative; // overfrow用
  height: 1000px;     // ページの高さを設定
  overflow: hidden;
}
```

MEMO

PC向けに同様のことを行いたい場合は、body要素だけでも指定可能です 01。

01 PCで横幅を画面にフィットさせる

```
body {
  overflow-x: hidden;
}
```

03

各ページをCSSで画面右外へ配置します 03-1。leftプロパティの値を使って、ページを左右に移動させます。

03-1 各ページを画面右側に配置する

```
section {
  position: absolute;
  left: 100%; // 基準位置（画面外へ置く）
  width: 100%;
}
```

リストナビゲーションの作成

04

リストナビゲーションをCSSで作ります。
隣接セレクタ（+）か、間接セレクタ（~）を利用して、リストの間のボーダーを指定します 04-1。

04-1 リストの間のボーダーを指定する

```
// 隣接セレクタの場合
nav li + li {
  border-top: 1px solid #bbb;
}
// 間接セレクタの場合
nav li ~ li {
  border-top: 1px solid #bbb;
}
```

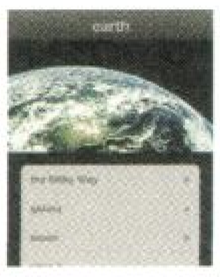
MEMO

リストの間にボーダーを引きたい場合、:first-child擬似クラスを利用しても同様の表示が可能です。
:first-childは、子要素の先頭要素を特定することができるので、まずすべての要素にボーダーを指定し、その後で先頭要素のボーダー指定を解除します 01。

01 :first-child擬似クラスの使用例

```
// すべてのli要素にボーダーを設定
nav li {
  border-top: 1px solid #bbb;
}
```

```
// 先頭要素のみ除外
nav li:first-child {
  border-top: none;
}
```



*COLUMN

隣接セレクタと間接セレクタ

隣接セレクタ（E＋F）01は、E要素とF要素が兄弟関係にあり、かつE要素のすぐ次に現れたF要素が対象になります。
間接セレクタ（E～F）02は、E要素とF要素が兄弟関係にあり、E要素以降に現れるF要素が対象になります。
03と04のような構造だと、隣接セレクタと間接セレクタの違いが出ます。

01 隣接セレクタ

```
// CSS
li + li {
  background: red;
}
// HTML
<ul>
  <li>list 1</li>
  <li>list 2</li> // 背景が赤色になる
  <li>list 3</li> // 背景が赤色になる
</ul>
```

02 間接セレクタ

```
// CSS
li ~ li {
  background: red;
}
// HTML
<ul>
  <li>list 1</li>
  <li>list 2</li> // 背景が赤色になる
  <li>list 3</li> // 背景が赤色になる
</ul>
```

03 隣接セレクタを使用した例

```
// 隣接セレクタ
// CSS
h2 + p {
  background: blue;
}
// HTML
<h2>見出し 1</h2>
<p>テキスト</p> // 背景が青色になる
<p>テキスト</p>
<h2>見出し 2</h2>
<p>テキスト</p> // 背景が青色になる
<p>テキスト</p>
```

04 間接セレクタの使用例

```
// 間接セレクタ
// CSS
h2 ~ p {
  background: blue;
}
// HTML
<h2>見出し 1</h2>
<p>テキスト</p> // 背景が青色になる
<p>テキスト</p> // 背景が青色になる
<h2>見出し 2</h2>
<p>テキスト</p> // 背景が青色になる
<p>テキスト</p> // 背景が青色になる
```

JavaScriptとtransitionプロパティでスライドを実装する

05

transitionプロパティを使って、ページを移動させます05-1。transition-durationで時間を指定すると、値が変更される際にアニメーションするようになります。

05-1 max-widthで最大画面幅を指定

```
section {
  -webkit-transition-duration: 600ms;
  -moz-transition-duration: 600ms;
}
```

*COLUMN

transitionプロパティ

transitionでは、要素の値を変更する際のアニメーション（トランジション）を指定することができます 01。

01 transitionプロパティの要素

プロパティ名	機能
transition-property	適用するプロパティ名を指定することで、アニメーションするプロパティを限定させることができます。カンマ区切りで複数指定も可能です
transition-duration	アニメーションにかける時間を指定します。単位はs（秒）かms（ミリ秒）で、0の場合でも単位は必要です。カンマ区切りで複数指定することで、transition-propertyで指定した複数プロパティそれぞれの時間を指定することができます。transition-propertyで指定したプロパティ数より少ない場合は、繰り返し扱いになります。02と03は同じ効果です。
transition-timing-function	アニメーションの加速や減速などのイージングを指定します。easeがデフォルトで、他にlinear、ease-in、ease-out、ease-in-outがあります
transition-delay	アニメーションが開始されるまでの待ち時間を指定します

02 時間を指定する値がtransition-propertyで指定したプロパティ数より少ない

```
transition-property: opacity, left, top, height;
transition-duration: 3s, 5s;
```

03 時間を指定する値がtransition-propertyで指定したプロパティ数と同じ

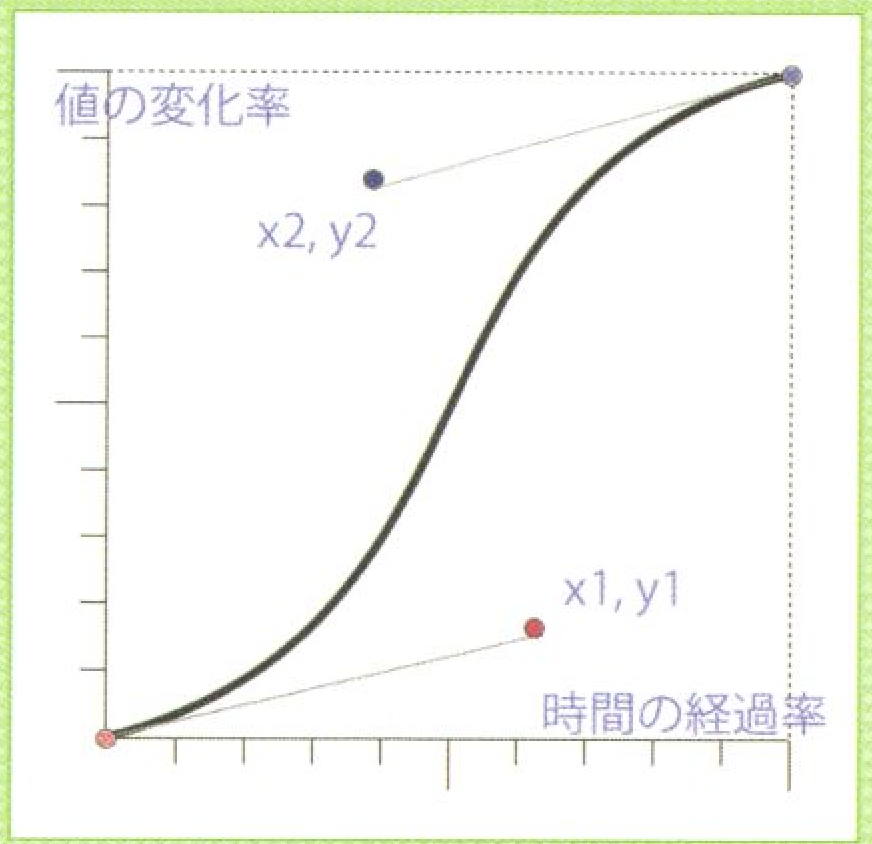
```
transition-property: opacity, left, top, height;
transition-duration: 3s, 5s, 3s, 5s;
```

*COLUMN

transition-timing-function

transition-timing-functionは、cubic-bezier()関数（3次ベジェ曲線）でイージングを指定することもできます。
3次ベジェ曲線は01のようなカーブで、値の変化していく様子を変更できます。

01 3次ベジェ曲線



02 cubic-bezier()関数の文法

```
cubic-bezier(x1, y1, x2, y2);
```

03 イージングの例

```
// 例
section {
  -webkit-transition-timing-function: cubic-bezier(0.42, 0, 0.52, 1);
  -moz-transition-timing-function: cubic-bezier(0.42, 0, 0.52, 1);
}
```

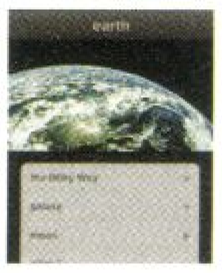
06

値の変化を開始するためには、何かきっかけが必要になります。
今回は、リストのリンクをクリックするとbody要素のクラス名を変更するようにします 06-1。
これはCSSではできないので、JavaScriptを使用します（簡易な記述になるよう、jQueryを用いています）。

06-1 リストのリンクをクリックすると、body要素のクラス名を変更する

```
// ドキュメントが読み込まれたら実行。
$(function() {

  // メニューをクリックした時の動作を指定。
  $("nav a").click(function(event) {
    // リンクのハッシュをもとに、bodyのクラス名を切り替え。
    var page = this.hash.slice(1);
    $("body").removeClass().addClass(page);
    // デフォルトのリンク遷移をキャンセル。
    event.preventDefault();
  });
});
```



07

各ページのIDと同じクラス名がbodyにあてられたときが、ページが呼ばれたときということになります 07-1 07-2。

07-1 2ページ目が呼ばれたとき

```
<body class="page-2">

  // 2ページ目
  <section id="page-2"> ... </section>

</body>

// CSS
.page-2 #page-2 { ... }
```

07-2 3ページ目が呼ばれたとき

```
<body class="page-3">

  // 3ページ目
  <section id="page-3"> ... </section>

</body>

// CSS
.page-3 #page-3 { ... }
```

08

目次となる初期ページ用に、bodyの最初のクラス名は「page-0」としておきます 08-1。

08-1 初期ページの設定

```
<body class="page-0">
  ...
</body>
```

09

各ページが呼ばれたときの設定を記述します 09-1。
これでリンクをクリックすると、ページが右から左へ動くようになりました。

09-1 各ページが呼ばれたとき

```
// 初期ページ（目次）
// クラス名がpage-0ではないとき（＝他の
ページを呼んだとき）
// 画面左に移動させる。
#page-0 { left: -100%; }
// 初期ページが呼ばれたとき（目次に戻って
きたとき）
// 画面内に表示されるようにする。
.page-0 #page-0 { left: 0; }

// 1ページ目が呼ばれたとき
.page-1 #page-1 { left: 0; }
// 2ページ目が呼ばれたとき
.page-2 #page-2 { left: 0; }
// 3ページ目が呼ばれたとき
.page-3 #page-3 { left: 0; }
```

ヘッダーを作成する

10

次に、「戻るボタン」を作成しましょう。その前に、まず「戻るボタン」を配置するヘッダーを作成します 10-1。

目次と同様に、リンクをクリックするとbody要素のクラス名をpage-0に切り替えられるように、nav要素の中のa要素を設定します。

10-1 ヘッダーを作る

```
<header>
  <h1>見出し</h1>
  <nav><a href="#page-0">back</a></nav>
</header>
```

11

ヘッダーの背景を、グラデーションで作成します 11-1。

11-1 ヘッダーの背景

```
header {
  background-image: -moz-linear-gradient(top, #666, #333);
  background-image: -webkit-linear-gradient(top, #666, #333);
}
```

ATTENTION!

グラデーションの指定の仕方は、01の通りです。中間色は省略可能で、位置は%で指定します。ベンダープレフィックスが必要です02。Chrome やSafari では、03のような指定も可能です。中間色は省略可能で、位置は%か0～1の小数値で指定します。

01 グラデーションの指定

```
linear-gradient(方向, 開始色, 中間色 位置(%), 終了色);
```

02 ベンダープレフィックス

```
-moz-linear-gradient();
-webkit-linear-gradient();
```

03 Chrome やSafari で可能な記述

```
-webkit-gradient(linear, 開始位置, 終了位置, from(開始色), color-stop(位置, 中間色), to(終了色));
```

12

12-1のようなボタンを作る際に、左側の三角部分もCSSのみで作成することができます。

まず、:before擬似要素で空要素を追加します12-2。次に、追加した空要素をtransformプロパティで回転させます。このとき、transform-originプロパティで、回転の基点となる位置を中心（デフォルト）から左上に変更しておきます12-3。

12-1 「戻るボタン」



12-2 :before擬似要素で空要素を追加する

```
header nav:before {
  content: "";
  width: 20px;
  height: 20px;
  position: absolute;
}
```

12-3 追加した空要素を、transformプロパティで回転させる

```
header nav:before {
  -moz-transform-origin: 0 0;
  -webkit-transform-origin: 0 0;
  -moz-transform: rotate(45deg);
  -webkit-transform: rotate(45deg);
}
```

*COLUMN

transformプロパティ

transformプロパティでは、要素に対して01のような変形を指定することができます。変形内容は、空白区切りで複数指定できます02。ただし、記述の順番で表示結果が異なりますので注意してください。例えば、03のコードでは、回転を行うと軸も一緒に回転するので、移動の後に回転するのか、回転の後に移動するのかで表示結果が異なります。

02 変形内容を複数指定できる

```
transform: rotate(30deg) translate(10px, -5px);
```

03 回転と移動の順番により結果が異なる

```
transform: rotate(30deg) translate(10px, -5px);
transform: translate(10px, -5px) rotate(30deg);
```

01 transformプロパティで指定できる変形

変形の種類	機能
rotate（角度）	回転。角度の単位はdeg。rotate(45deg)で45度回転
scale（倍率）	拡大/縮小。倍率は数値で指定、カンマで区切ることでX軸Y軸それぞれを指定することができます。scale(0.5, 1.2)でX軸方向に0.5倍、Y軸方向に1.2倍
skew（角度）	歪み（シアー）。角度の単位はdeg。skew(5deg)で5度歪ませます
translate（距離）	移動。距離の単位はpx。カンマで区切ることでX軸Y軸それぞれを指定することができます。translate(30px)でX軸Y軸方向それぞれ30px分移動させます

MEMO

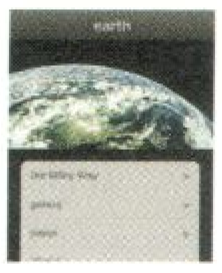
transform-originプロパティでは、transformプロパティで行う変形の基準点を指定することができます01。デフォルトは、要素の中心位置（50%, 50%）です。位置の指定はbackground-positionプロパティなどと同様です02。

01 transform-originプロパティ

```
transform-origin: X軸 Y軸;
```

02 位置指定の記述

```
transform-origin: left center;
transform-origin: 20px 0;
```



ヘッダーの内容をページごとに切り替える

13

bodyにあてられたクラス名を利用することで、ヘッダーに表示される見出しをページごとに変更することもできます。
:after擬似要素とcontentプロパティを使って、title属性の値を見出しとして表示させます 13-1。

13-1 title属性の値を見出しとして表示する

```
// HTML
<header>
  <h1 title="page0 title"></h1>
  <nav><a href="#page-0">back</a></
  nav>
</header>

// CSS
.page-0 header h1:before {
  content: attr(title);
}
```

MEMO

contentプロパティでは、attr()関数を使って要素の属性値を表示できます 01。

01 attr()関数を使って要素の属性値を表示

```
// HTML
<p title="abc">DEF</p>

// CSS
p:before {
  content: attr(title);
}

// 表示結果
abcDEF
```

14

各ページごとに見出しを変更できるよう、独自データ属性を利用します。
独自データ属性は、カスタムデータ属性とも呼ばれ、「data-名前="値"」の形式でユーザが自由に属性と値を設定することができます。
ページごとの見出しと、スタイル記述を追加します 14-1。

14-1 attr()関数を使って要素の属性値を表示

```
// HTML
<header>
  <h1 title="page0 title" data-page1-
  title="page1 title" data-page2-
  title="page2 title" data-page3-
  title="page3 title"></h1>
  <nav><a href="#page-0">back</a></
nav>
</header>
```

```
// CSS
.page-1 header h1:before {
  content: attr(data-page1-title);
}
.page-2 header h1:before {
  content: attr(data-page2-title);
}
.page-3 header h1:before {
  content: attr(data-page3-title);
}
```

ATTENTION!

独自データ属性の「名前」の部分は、ほかの属性名の命名規則と同じく、小文字のアルファベットか数字で構成された文字列をハイフンで区切ることができます。ただし、文字列の先頭は小文字のアルファベットでなければなりません。01

01 独自データ属性の書式

独自データ属性（data-名前="値"）

まとめ [CONCLUSION]

CSS3で用意されるtransitionやanimationなどのアニメーションは、JavaScriptを使用したものよりも描画がスムーズに行われるようにできています。
また、多様なプロパティを利用することで、画像を用意しなくても多彩な表現が可能になり、転送にかかるファイルサイズを抑制することが可能になります。
これらは、スマートフォンなどPCと比較して低スペックなマシンや、通信速度が限られた環境では特に有効になります。単純なアニメーションなどでは、積極的に活用してみてください。

09 カスタムURLスキームと文字量の変化に対応できるリスト



スマートフォンでは「カスタムURLスキーム」と呼ばれる、アプリケーションへ直接リンクできるURLがあります。ブラウザからの「カスタムURLスキーム」の使い方と、一風変わった柔軟性のあるリスト表示を交えてそれらを紹介します。

制作・文／執行大介

制作のポイント

- 「display:table;」と「table-layout: fixed;」を使ったリスト
- メディアクエリによる端末の横向き時の表示の切り替え
- 「tel:」などに代表される「カスタムURLスキーム」の使い方

対応ブラウザ

iOS	3.0 以上
Android	2.1 以上

※Androidは一部対応していない機種があります。

TECHNIC 1

複数行の中央揃えが可能

「display:table-cell;」を使うことで、リストに入れる要素の幅を気にせず複数行の中央揃えを実現することができます。



TECHNIC 2

アプリへのスムーズなリンク

「カスタムURLスキーム」を使うことで、従来使われてきたメールへのリンク（mailto:）だけにとどまらず、スマートフォンのアプリへのスムーズなリンクを実装することができます。





ベースとなるHTMLの記述

01

まずはHTMLにとりかかります。01-1のコードの「<div class="mainSec">…</div>」内に01-2のように記述していきます。はじめに「お問い合わせ」の欄をマークアップしていきます。クラス「contactListInner」の中にリスト（li）タグでanchorタグ（以下aタグ）を囲んだものを3つ記述し、のちほどaタグの「href要素」に「カスタムURLスキーム」を指定します。

01-1 ベースとなるHTML（index.html）

```
<!DOCTYPE HTML>
<html lang="ja">
<head>
<meta charset="UTF-8">
<title>便利なアプリケーションへのリンクと柔軟性のあるリスト</title>
<meta name="keywords" content="">
<meta name="description" content="">

<link rel="index" href="/">

<meta name="format-detection" content="telephone=no">
<meta name="viewport" content="width=device-width, user-scalable=yes, initial-scale=1, maximum-scale=1"/>
<meta name="apple-mobile-web-app-capable" content="yes">

<link rel="stylesheet" href="css/common.css">
<link rel="stylesheet" href="css/url_scheme.css">

</head>

<body>
<header>
<h1 class="globalHeaderLogo"><img src="images/logo.png" alt="Books Panda Space"></h1>
</header>

<article class="mainArticle">

<section class="mainHead">
<h1>■アプリケーションへのリンク</h1>
<p>いろいろなアプリへリンクします。</p>
</section>

<div class="mainSec">
// ここに01-2の内容を記述
<!-- / .mainSec --></div>
<!-- / .mainArticle --></article>

<footer class="globalFooter">
<p class="copyringht">Books Panda Space</p>
<!-- / .globalFooter --></footer>
</body>
</html>
```

01-2 <div class="mainSec">内に記述する内容（index.html）

```
<div class="contact">
<h2>お問い合わせ先</h2>

<div class="contactList">
<section class="contactListInner">
<h3>メール</h3>
<ul>
<li><a href="">店舗に関して</a></li>
<li><a href="">商品に関して</a></li>
<li><a href="">採用に関して</a></li>
</ul>
</section>

<section class="contactListInner">
<h3>電話</h3>
<ul>
<li><a href="">店舗に関して</a></li>
<li><a href="">商品に関して</a></li>
<li><a href="">採用に関して</a></li>
</ul>
</section>
<!-- / .contactList --></div>
<!-- / .contact --></div>
```

MEMO

今回のサンプルでは、2つのCSSファイルを読み込んでいます。「common.css」は全般的なスタイルの設定、「url_scheme.css」は個別要素に対するスタイルの設定を行っていますが、本書では「url_scheme.css」の内容だけを解説します。

02

次に標準のアプリへ連携させる欄 02-1 と、サードパーティー製のアプリを連携させる欄 02-2 です。ulタグにクラス「appleList」という名前をつけ、01 と同じようにリストタグの中にaタグを記述し、こちらはアイコンを「figure」タグ、タイトルを「h2」タグで記述しました。

02-1 標準のアプリへ連携させる欄（index.html）

```
<ul class="appleList">
<li><a href="">
<figure><img src="images/no2_icon_03.png" alt=""></figure>
<h2>SMSを書く</h2>
</a></li>

<li>
<a href="">
<figure><img src="images/no2_icon_04.png" alt=""></figure>
<h2>Mapを見る<span class="toggleBR"><br></span>：場所を検索</h2>
</a></li>

<li>
<a href="">
<figure><img src="images/no2_icon_04.png" alt=""></figure>
<h2>Mapを見る<span class="toggleBR"><br></span>：経路を検索</h2>
</a></li>
<!-- / .appleList --></ul>
```

02-2 サードパーティー製のアプリを連携させる欄（index.html）

```
<ul class="appleList">
<li><a href="">
<figure><img class="twitter" src="images/no2_icon_05.png"
alt=""></figure>
<h2>Twitterに投稿</h2>
</a></li>

<li>
<a href="">
<figure><img src="images/no2_icon_06.png" alt=""></figure>
<h2>Facebook<span class="toggleBR"><br></span>：投稿する</h2>
</a></li>

<li>
<a href="fb://feed">
<figure><img src="images/no2_icon_06.png" alt=""></figure>
<h2>Facebook<span class="toggleBR"><br></span>：Feedを見る</h2>
</a></li>
<!-- / .appleList --></ul>
```

CSSによるスタイル設定

03

次にCSSを記述していきます。まずは「お問い合わせ先」を囲む「contact」というクラスのCSSです 03-1。大きさを指定したあと、「box-sizing: border-box;」でボーダー（paddingも）の幅をwidthの大きさに含めないようにします。この指定をすることで、「width:100%;」を指定した場合、ボーダーやパディングの幅にかかわらず親要素の100%の幅を確保できます。さらに「box-shadow」に内側の影と外側の影をつけます。カンマ区切りで複数指定することで、同時に影を表示できます。

03-1 「お問い合わせ先」全体を囲む要素（url_scheme.css）

```
.contact{
  /*大きさと下部のマージンを設定*/
  position:relative;
  margin:0px 0px 22px;
  width:100%;

  /*ボーダーをwidthの大きさに含めないようにする*/
  -webkit-box-sizing: border-box;
  box-sizing: border-box;

  background-color:#FFF;

  /*内側の影と外側の影をつける*/
  -webkit-box-shadow: inset 0px 1px 1px rgba(0, 0, 0, 0.6), 0px 1px 1px #e6eaef;
  box-shadow: inset 0px 1px 1px rgba(0, 0, 0, 0.6), 0px 1px 1px #e6eaef;

  -webkit-border-radius: 6px;
  border-radius: 6px;
  border:#ababab 2px solid;
}
```



04

次に「お問い合わせ先」の見出しの部分です 04-1 04-2。「padding」で文字の余白を指定し、「width」で幅を指定しますが、03と同じく「box-sizing: border-box;」を指定しています。以後も、スマートフォンを横に向けた場合も幅100%を確保できるように逐一入れていきましょう。さらにiPhoneライクな見出しにするため、「text-shadow」と「linear-gradient」を指定しています。ここでは見出しのテキストにボールドを指定していませんが、Androidではボールド指定しても効かないので、注意が必要です。

04-1 「お問い合わせ先」の見出し（url_scheme.css）

```
.contact h2{
  padding:10px;
  width:100%;

  /*ボーダーをwidthの大きさに含めないようにする*/
  -webkit-box-sizing: border-box;
  box-sizing: border-box;

  /*iPhoneライクなテキストの指定*/
  color:#fff;
  text-shadow:0px -1px 1px rgba(0,0,0,.5);

  /*見出しの背景グラデーション*/
  background-image: -webkit-gradient(linear, left top, left bottom, color-stop(0%, #b0bccc), color-stop(50%, #889bb3), color-stop(50%, #8094ae), color-stop(100%, #6d84a2));
  background-image: -webkit-linear-gradient(-90deg, #b0bccc 0%, #889bb3 50%, #8094ae 50%, #6d84a2 100%);
  background-image: linear-gradient(-90deg, #b0bccc 0%, #889bb3 50%, #8094ae 50%, #6d84a2 100%);

  /*上下のボーダーでよりiPhoneらしくする*/
  border-top:#cdd5df 1px solid;
  border-bottom:#2d3642 1px solid;
}
```

04-2 「お問い合わせ先」の見出し



05

次に「メール」と「電話」のそれぞれのセクションの記述です 05-1。div要素の「contactList」というクラスの中に、「contactListInner」というクラスのセクションが横に50%ずつの大きさで並びます。このような場合は親要素に「display:table;」と「table-layout:fixed;」を指定し、子要素に「display:table-cell;」と記述すると、表組みのように横幅が変わっても子要素は50%を維持できるレイアウトが可能です。さらに、「メール」と「電話」のボックスの境界線を「電話」側の「border-right」で表現しました。

05-1 「メール」と「電話」のセクション（url_scheme.css）

```
.contactList{
  /*内包する要素をテーブルのように並べる*/
  display:table;
  /*内包する要素が合計で横幅100%になる*/
  table-layout:fixed;

  width:100%;

  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}

.contactListInner{
  display:table-cell;
}
.contactListInner:first-child{
  /*横並びのボックスの縦のボーダー*/
  border-right:1px #666 solid;
}
```

06

次の「メール」と「電話」の見出し部分も04と同じようにiPhoneライクな表現ですが、こちらはグラデーションを異なる表現にしました06-1 06-2。

06-2 「メール」と「電話」の見出し部分



06-1 「メール」と「電話」の見出し部分（url_scheme.css）

```
.contactListInner h3{
  padding:6px;

  color:#fff;
  text-shadow:0px -1px 1px rgba(0,0,0,.5);

  /*04のグラデーション違い*/
  background-image: -webkit-gradient(linear, left top, left bottom, color-stop(0%,
  #92a0ab), color-stop(100%, #b6bec5));
  background-image: -webkit-linear-gradient(-90deg, #92a0ab 0%, #b6bec5 100%);
  background-image: linear-gradient(-90deg, #92a0ab 0%, #b6bec5 100%);

  border-top:#717d85 1px solid;
  border-bottom:#989ea4 1px solid;
}
```

07

次に各リスト要素の記述です07-1。「border-bottom」で各リストの下にボーダーをつけ、「:last-child」セレクタを使い最後の子要素のみボーダーをなくすようにしました。さらに子要素のaタグに「display: table-cell;」を指定することで、「vertical-align: middle;」を使えるようにしています。これで、内包するテキスト要素が複数行になっても縦中央位置を確保できます。

07-1 リスト要素（url_scheme.css）

```
.contactListInner li{
  padding:0px 12px;
  font-size:12px;

  /*各リストの下のボーダー*/
  border-bottom: 1px solid gray;

  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}
/*最後の子要素は下のボーダーをなくす*/
.contactListInner li:last-child{
  border-bottom:none;
}

.contactListInner li>a{
  display: table-cell;
  padding: 0px 0px 0px 37px;

  width:100%;
  height: 45px;

  /*文字を縦中央位置に指定する*/
  vertical-align: middle;

  background-position:left center;
  background-repeat:no-repeat;
  /*アイコンのサイズを指定*/
  -webkit-background-size: 30px 31px;
  background-size: 30px 31px;
}
```

08

さらにアイコンの表示ですが、「お問い合わせ先」のリンクは背景画像でアイコンを表示しています。ここでは3つのaタグから同じアプリケーションに連携させているので、属性セレクタの条件であるE[foo^="bar"]を使うことにより、もしリンク先で使いたいアプリケーションが変更されてもHTML側では更新の必要はありません。ここでは「href要素が○○で始まるとき」という指定で「[href^="mailto"]」と記述しています08-1。

MEMO

属性セレクタの条件には3種類あります。「^」は指定した条件で始まっている場合、「$」は指定した条件で終わっている場合、「*」は指定した条件が含まれている場合です。



08-1 アイコンの表示（url_scheme.css）

```
/*属性セレクタで表示するアイコンを変える*/
.contactListInner li>a[href^="mailto"]{
  background-image: url(../images/no2_icon_01.png);
}
.contactListInner li>a[href^="tel"]{
  background-image: url(../images/no2_icon_02.png);
}
```

09

続いて「SMSを書く」以下のリストですが、大まかなCSSは「お問い合わせ先」の中身とほぼ同じ作りです 09-1 09-2。こちらはアイコンをimgタグで表示していますが、背景画像で指定しても構いません。ここではリストのテキストが長いものがあり、その改行を「@media screen and (orientation: landscape)」でコントロールしています。スマートフォンを横向きで見ているときは「orientation:landscape」となり、改行タグを囲んでいるspanタグが表示されず、縦で見ているときは「orientation:portrait」となるので、任意の位置で改行できる仕組みです。

09-1 「SMSを書く」以下のリスト（url_scheme.css）

```
/*メディアクエリを使い端末の向きで改行を
調整*/
@media screen and (orientation:landscape) {
  .toggleBR{
    display:none;
  }
}
@media screen and (orientation:portrait) {
  .toggleBR{
    display:inline;
  }
}
```

09-2 「SMSを書く」以下のリスト



カスタムURLスキームの記述

10

最後にカスタムURLスキーム 10-1 を記述します。URLスキームの後に最初の値を記述し、最初のオプションの前に「?」を入れ「=」で要素を指定します。2個以上オプションがある場合は「&」で連結しますが、リンクさせるアプリケーションによって記述方法が微妙に異なります。以下で一部の記述例を紹介しますが、詳しくはサンプルファイルを参考にしてください。

10-1 カスタムURLスキームの種類

アプリ名	URLスキーム	最初の値（例）	オプション（例）	オプションの意味
メール	mailto:	宛先のアドレス	cc	CCの宛先
			bcc	BCCの宛先
			subject	メールの件名
			body	メールの本文
電話	tel:	電話番号	—	—
SMS / MMS	sms:	宛先のアドレス	—	—
Google Maps *	maps:	特になし	q	検索する場所
			daddr	出発地点
			saddr	到着地点
Twitter *	twitter:	//post	message	投稿の本文
Facebook *	fb:	//publish/	text	投稿の本文
		//feed	—	フィードを見る

*のついたカスタムURLスキームは、Android端末では機能しない場合があります。

11

たとえばクリックするとタイトルと本文つきのメールを立ち上げる場合は、「mailto:」の後に送信させたいアドレス、その後に「?subject=タイトル」、その後に「&body=本文」となります。また日本語での指定でも動きますが、極力URI形式へエンコードして記述しましょう。この場合のカスタムURLスキームは「mailto:送信させたいアドレス?subject=タイトル&body=本文」のようになります。この日本語部分をURI形式へエンコードして、href要素に記述しましょう。
CCやBCCのアドレスを指定する場合は、11-1のようになります。

11-1 カスタムURLスキームの記述例（メール）

```
<a href="mailto:test@tets.com?cc=shop@example.com&bcc=e_shop@example.com&subject=%E5%BA%97%E8%88%97%E3%81%AB%E9%96
%A2%E3%81%97%E3%81%A6&body=%E6%9C%AC%E6%96%87">店舗に関して</a>
```

12

もう1つの例としてGoogle Mapsへのリンクを書いてみましょう。場所をピンで指定するだけでなく、iPhoneの「現在地から目的地へ」の経路検索12-1もカスタムURLスキームから可能です12-2。まず「maps:」の後に到着地オプション「daddr」と到着地（東京タワー）の緯度と経度を「=」でつなぎます。さらに出発地オプション「saddr」に「現在地」と指定して（URI形式へエンコード）、同じく「=」でつなぎます。これで出来たカスタムURLスキーム「maps:daddr=35.658499,139.74553&saddr=%E7%8F%BE%E5%9C%A8%E5%9C%B0」をクリックするとマップが立ち上がり、現在地から目的地までの経路検索結果が表示されます。ただし、こちらは残念ながらAndroidではリンクされません。

12-1 iPhoneでの経路検索



12-2 カスタムURLスキームの記述例（経路検索）

```
<a href="maps:daddr=35.658499,139.74553&saddr=%E7%8F%BE%E5%9C%A8%E5%9C%B0">
<figure><img src="images/no2_icon_04.png" alt=""></figure>
<h2>Mapを見る<span class="toggleBR"><br></span>：経路を検索</h2>
</a>
```

まとめ [CONCLUSION]

「display:table;」と「table-layout:fixed;」、および「display:table-cell;」を使ったリストはメリットが2つあります。中の文字量を問わず縦中央位置にできる点と、横に並べることができる点です。実務ではデザインフィックス後に文字の量が変わることが多々ありますが、この方法だとそのような場合でも柔軟に対応できます。
また改行位置も、任意の位置で改行した場合改行をそのままにしておくと、筐体を横に向け全体の横幅が伸びた場合に余白が大きくなってしまいますが、紹介した方法を使えば横にしたときに改行を取ることができます。
今回のメディアクエリは特殊な使い方ですが、参考にしてください。さらにカスタムURLスキームはサイトのお問い合わせなどで従来使われてきた「mailto:」だけにとどまらず、スマートフォンならではのアプリを活用することでユーザーの反応をよくすることができるはずなので、ぜひ活用してください。

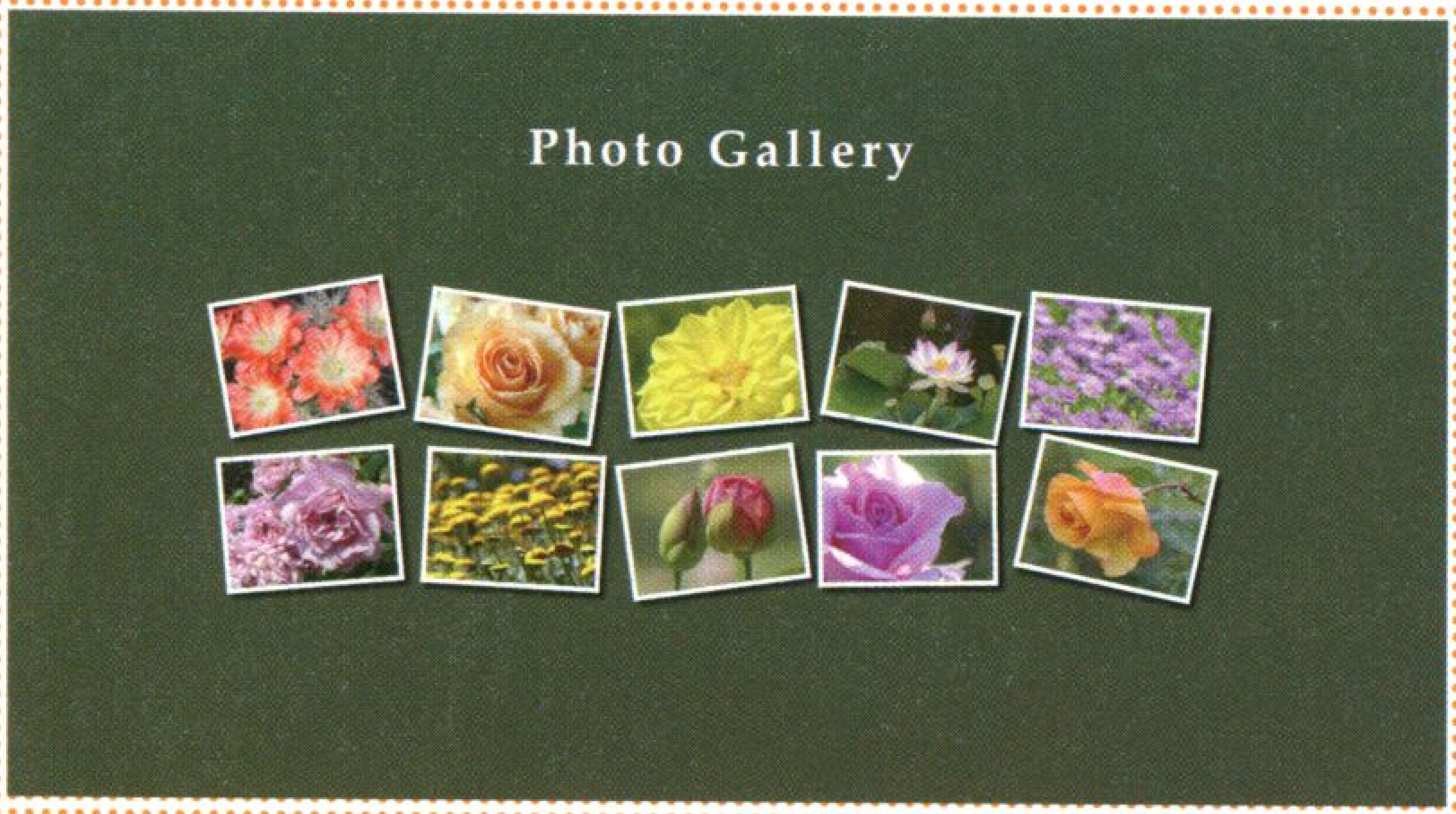
CHAPTER 03

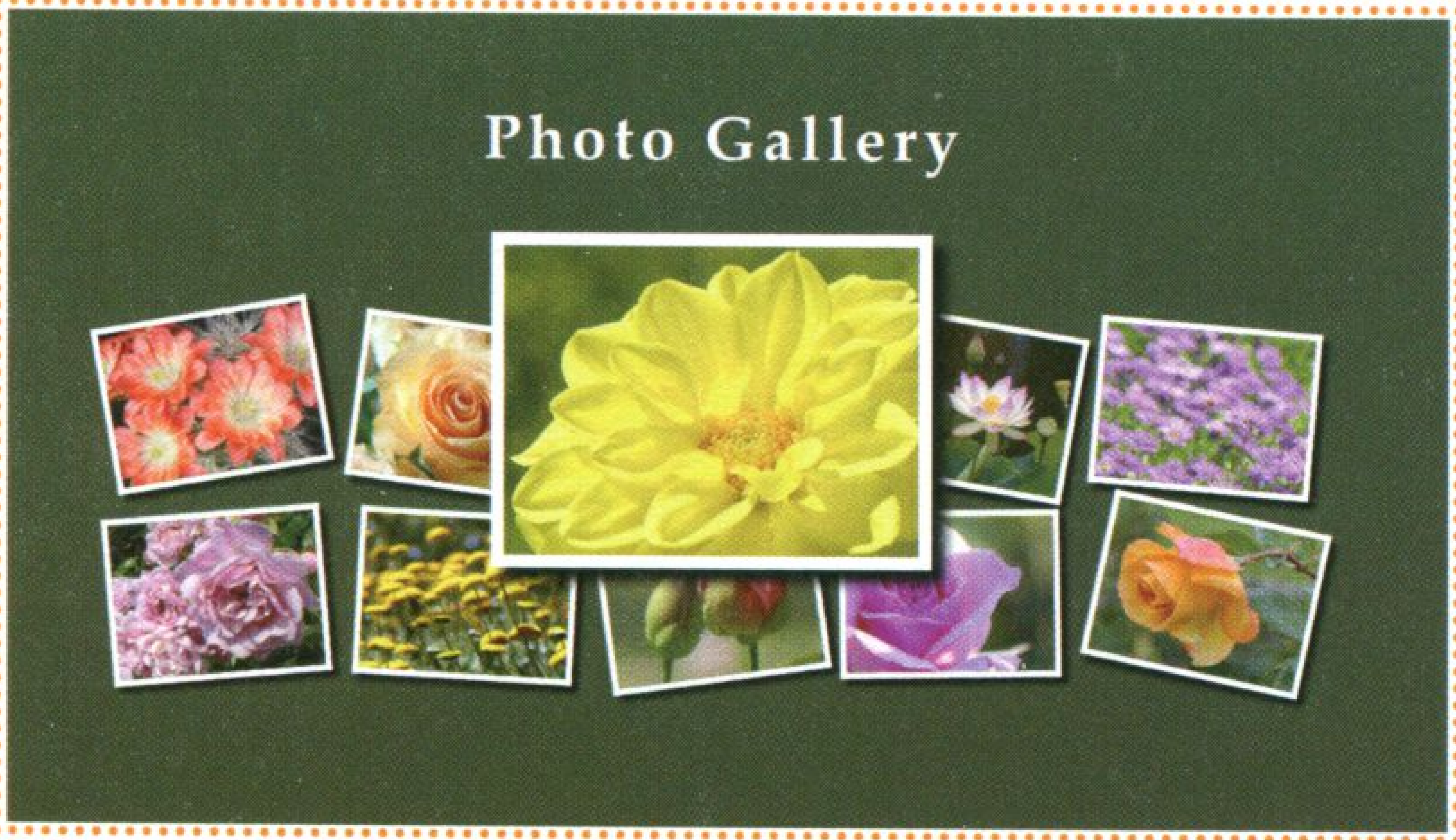
グラフィック・タイポグラフィ

Graphic & Typography

- インタラクションのあるフォトギャラリー
- border-imageを活用したポラロイド風のフレーム
- マスク効果を利用した印象的なグラフィック
- Webフォントでテキストを華やかに彩る
- 属性セレクタで写真を装飾したギャラリー
- text-shadowを使って袋文字や立体的な文字を表現
- 画像の重ね合わせで実現するさまざまな背景
- DropCapsを使った雑誌風のテキスト表現
- ドロップシャドウ効果で写真にアクセントをつける

01 インタラクションのあるフォトギャラリー





transformを用いて、サムネイル画像を効果的に見せるフォトギャラリーを作成します。
また、transformとtransitionの組み合わせにより、CSSのみでサムネイルがマウスに反応する効果を作成します。

制作・文／河原 覚（株式会社アークフィリア）

制作のポイント

- transformによる回転・拡大・移動
- transitionの利用
- プログレッシブ・エンハンスメントを考慮した設計

対応ブラウザ

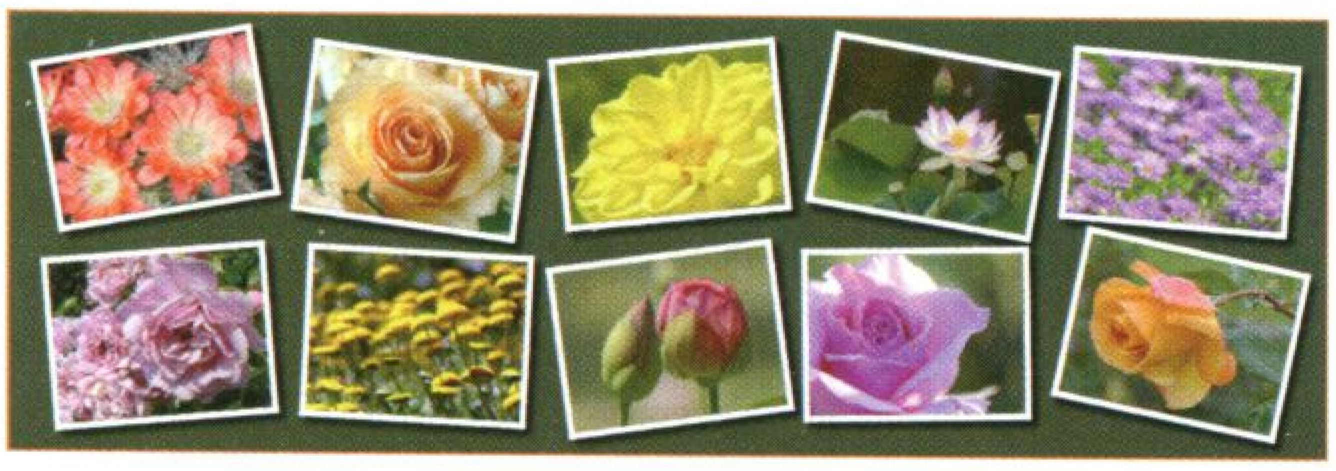
IE	P.E.
Safari	5以上
Firefox	8以上
Chrome	16以上

※IEではtransform、transitionの効果に対応していません。

TECHNIC 1

サムネイルの配置に動きをつける

transformを利用して散らばったようなレイアウトを作ります。

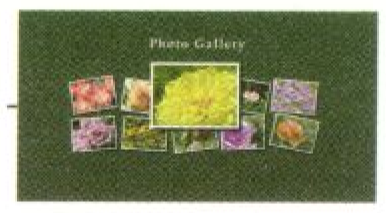


TECHNIC 2

transitionによるアニメーション

transitionによってtransformの変化をアニメーションさせます。





基本のレイアウトを作成する

01

サムネイルを作成します。本作例ではサムネイルに触れていないときは横160×縦120px、拡大時には横320×縦240pxとなるようにします。拡大後に画像を綺麗に見せるため、サムネイル画像は拡大後の大きさ（横320×縦240px）として書き出しておきます。
HTMLはulを使ってコーディングし、各サムネイルは後ほど個別に効果を設定するので、idを振っておきます 01-1 。

01-1 HTML

```
<ul id="Gallery">
  <li id="Photo01"><a href="#"><img src="images/photo01_thumb.jpg" alt=""></a></li>
  <li id="Photo02"><a href="#"><img src="images/photo02_thumb.jpg" alt=""></a></li>
— 省略 —
  <li id="Photo10"><a href="#"><img src="images/photo10_thumb.jpg" alt=""></a></li>
</ul>
```

02

CSSをコーディングしていきます。本作例では、装飾のためにbox-shadow、transform、transitionを利用していきますが、ブラウザのバージョンによって対応状況に差があります。
ここでは、対応状況に応じた4段階で、少なくとも表示が崩れないような設計を考えて進めていきます 02-1 。

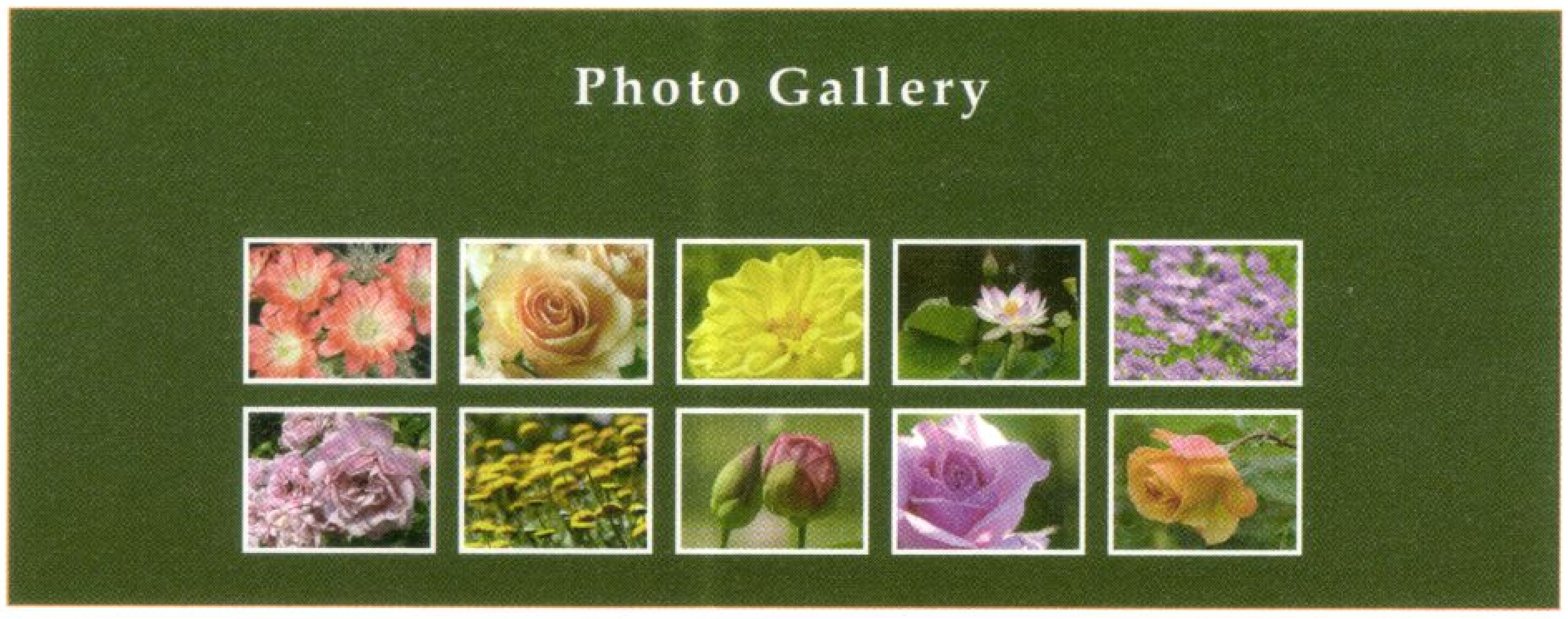
02-1 ブラウザのバージョンごとに対応する段階

対応状況	表示
css3に対応していない	単純なグリッド表示
box-shadowに対応	サムネイルに影がつく
transformに対応	ばらついた効果と、ホバー
transitionに対応	ホバー効果がアニメーション

03

まずは、サムネイルがグリッド状に並んでいるレイアウトになるようにコーディングしていきます 03-1 。
この状態をtransformに対応していないブラウザでの見た目とするため、このレイアウトではCSS3の新機能などは使わずに、floatを用いてコーディングを行います。liとimgのサイズは、ともに横160×縦120pxに固定しておきます 03-2 03-3 。

03-1 この段階のブラウザ表示



03-2 リスト

```
#Gallery li {
  float:left;
  width:160px;
  height:120px;
  margin:0 20px 20px 0;
  border:5px solid #ffffff;
}
```

03-3 画像

```
#Gallery li a img {
  width:160px;
  height:120px;
}
```

04

基本的なレイアウトができたら、拡大したときの重ねあわせを制御するために、あらかじめposition:relative;とz-indexの初期値を設定しておきます 04-1。

04-1 #Gallery li にz-indexの初期値設定を追加

```
position:relative;
z-index:10;
```

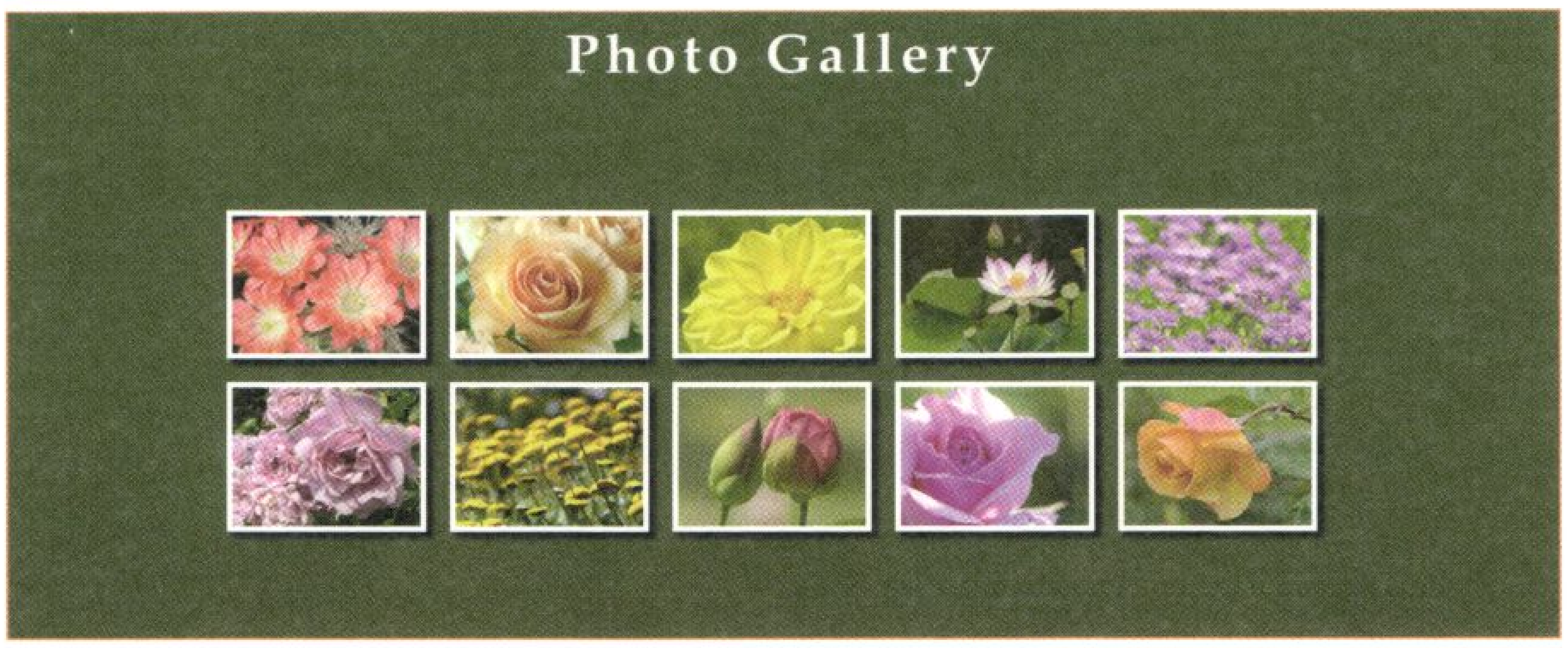
05

サムネイルの装飾を行っていきます。まずは、box-shadowで影を設定します 05-1。transformに対応していないが、box-shadowを描画できるブラウザでは、この状態になります 05-2。

05-1 #Gallery li にbox-shadowを追加

```
-moz-box-shadow:5px 5px 5px #000000;
-webkit-box-shadow:5px 5px 5px #000000;
box-shadow:5px 5px 5px #000000;
```

05-2 この段階のブラウザ表示



写真が散らばっているような効果を作る

06

次に、transformを用いて、写真が散らばっているような効果を作っていきます。
本作例では散らばっているような効果、およびマウスが触れた際の拡大効果のため、transformのscale(拡大)、rotate(回転)、translate（移動）を用います。それぞれの基本的な書式は 06-1 06-2 06-3 06-4 のようになります。
scaleには拡大率、rotateには角度、translateには横→縦の順で移動する距離を指定します。

06-1 scale

```
-moz-transform: scale(1.0);
-webkit-transform: scale(1.0);
transform: scale(1.0);
```

06-2 rotate

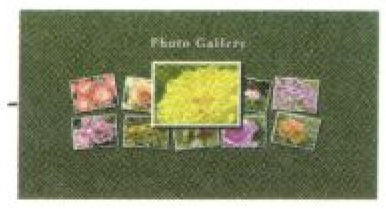
```
-moz-transform: rotate(0deg);
-webkit-transform: rotate(0deg);
transform: rotate(0deg);
```

06-3 translate

```
-moz-transform: translate(0,0);
-webkit-transform: translate(0,0);
transform: translate(0,0);
```

06-4 複数のオプションを指定する場合

```
-moz-transform: scale(1.0) rotate(0deg) translate(0,0);
-webkit-transform: scale(1.0) rotate(0deg) translate(0,0);
transform: scale(1.0) rotate(0deg) translate(0,0);
```



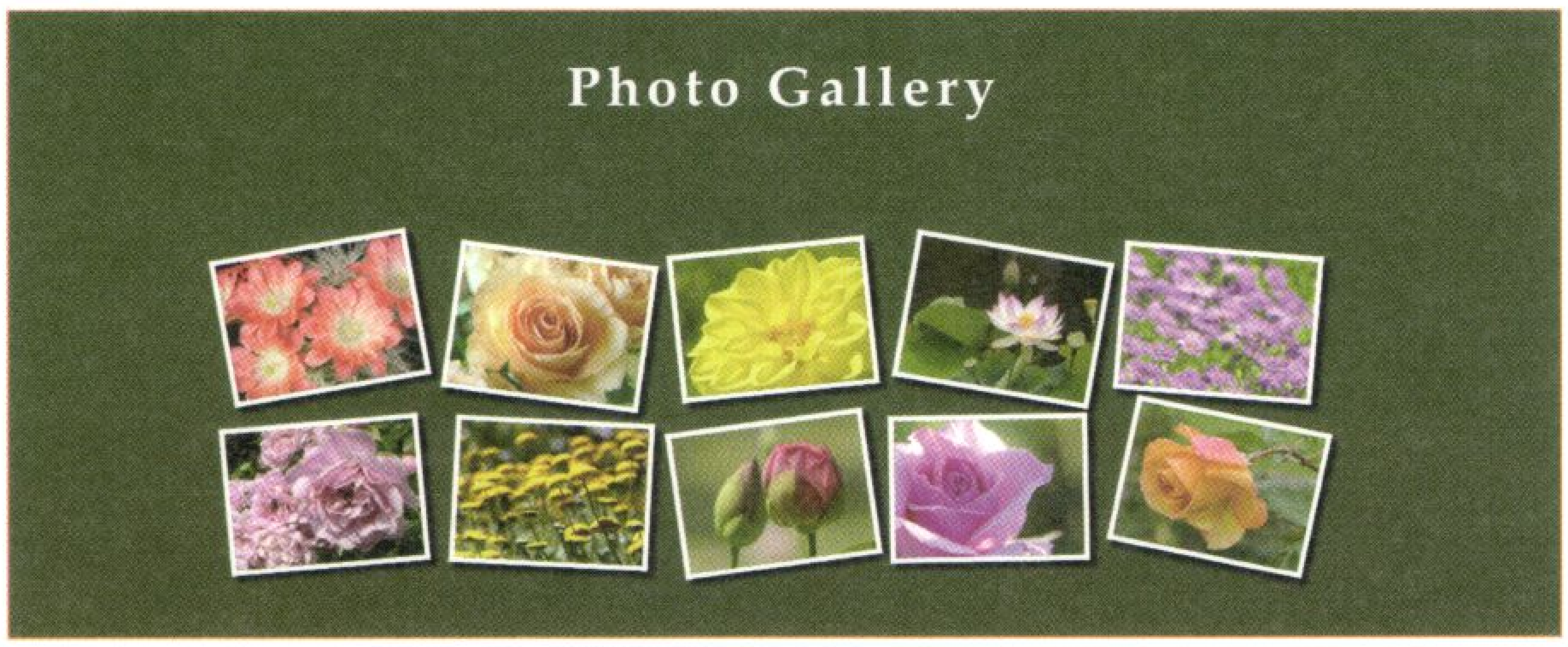
07

角度と位置のブレをrotate、translateを用いて設定していきます。散らばったような効果を出すため、あらかじめ振っておいたサムネイルのidに対して、1つずつ別の値を設定していきます。
値を設定する際は、次ステップの拡大効果のために、scale(1.0)も一緒に設定しておきます 07-1 07-2。

07-1 個別の効果

```
#Photo01 {
  -moz-transform: scale(1.0) rotate(-10deg) translate(-6px,-3px);
  -webkit-transform: scale(1.0) rotate(-10deg) translate(-6px,-3px);
  transform: scale(1.0) rotate(-10deg) translate(-6px,-3px);
}
```

07-2 散らばったような効果



マウスに反応する効果を作る

08

マウスに反応する効果を作っていきます。マウスがサムネイルに触れたときに、回転と移動を元に戻し、サムネイルを2倍に拡大するような効果を設定します。
liのhover擬似クラスにrotateとtranslateを0に戻し、scaleを2.0に設定するようなtransformを設定します 08-1。

08-1 拡大効果

```
#Gallery li:hover {
  -moz-transform: scale(2.0) rotate(0deg) translate(0,0);
  -webkit-transform: scale(2.0) rotate(0deg) translate(0,0);
  transform: scale(2.0) rotate(0deg) translate(0,0);
}
```

MEMO

transformを利用する際の回転や拡大の中心点は、無指定の場合、対象ブロックの横50%、縦50%の位置になります。中心点を変更したい場合はtransform-originを利用します 01

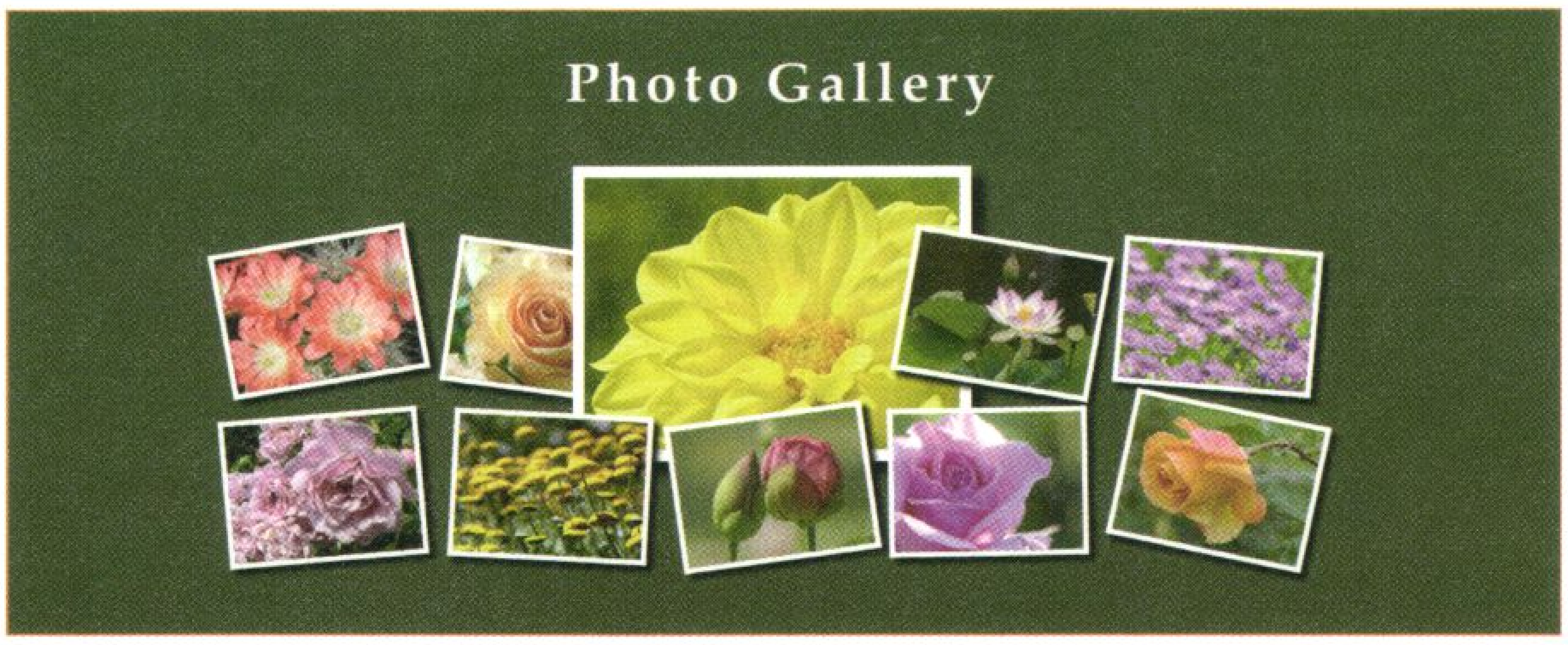
01 ブロックの右下を指定する例

```
-moz-transform-origin: 0 100%;
-webkit-transform-origin: 0 100%;
transform-origin: 0 100%;
```

09

これで触れたときの効果はできましたが、このままでは、拡大されたサムネイルが他のサムネイルの下に隠れてしまっています 09-1 。
そこで、hoverのスタイルにz-indexを設定します。ここに設定するz-indexは、04で指定したz-indexよりも大きいものにします 09-2 。
ここまでのスタイルが、transformに対応しているブラウザのものとなります 09-3 。

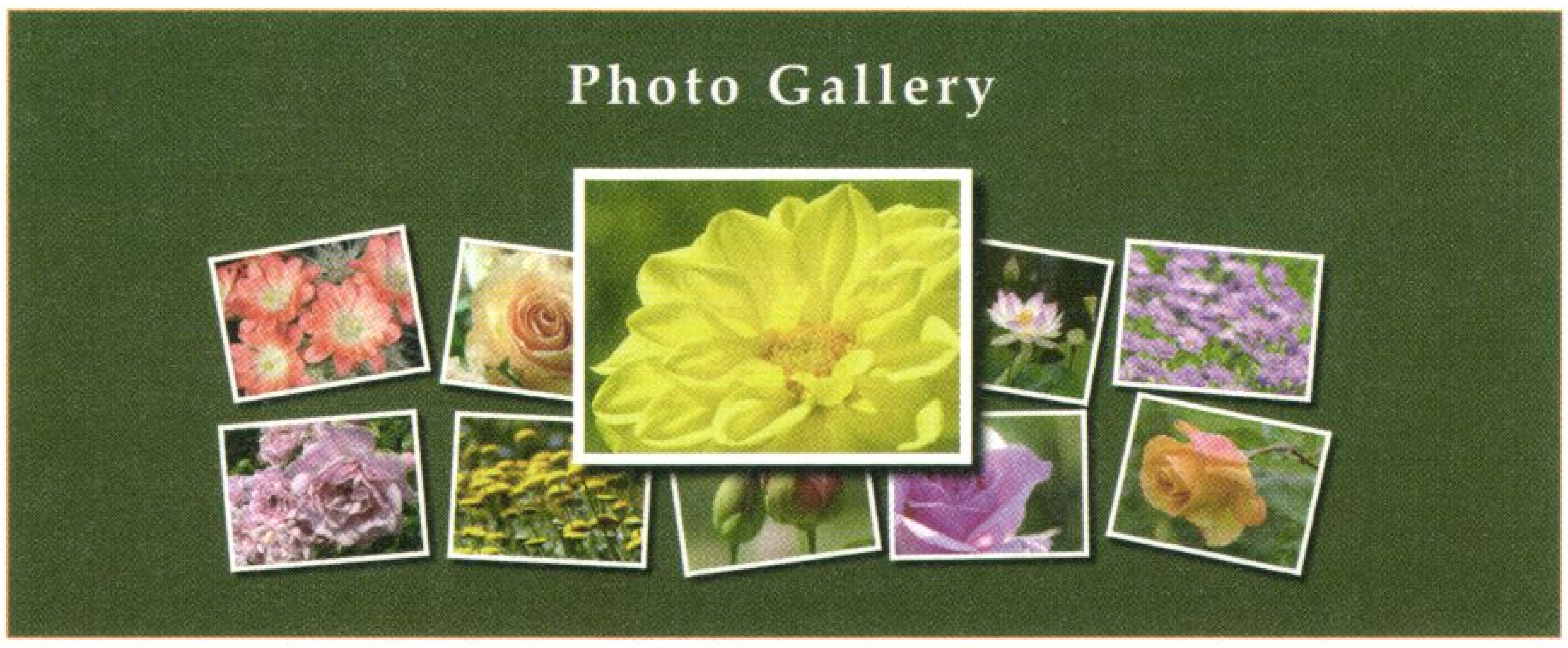
09-1 中央の画像が他のサムネイルに隠れている



09-2 z-indexの設定

```
#Gallery li:hover {
  -moz-transform: scale(2.0) rotate(0deg) translate(0,0);
  -webkit-transform: scale(2.0) rotate(0deg) translate(0,0);
  transform: scale(2.0) rotate(0deg) translate(0,0);
  z-index:100;
```

09-3 中央の画像が他のサムネイルよりも上になった



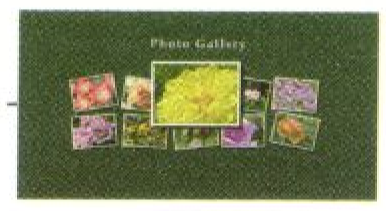
transition対応ブラウザ向けのアニメーション効果を設定する

10

最後に、transitionに対応しているブラウザ向けのアニメーション効果を設定します。
transitionは通常の状態からhover状態など、状態間のスタイルの違いをアニメーションさせるためのスタイルです。サムネイルの各状態に設定されたtransformスタイルの違いを、transitionによってアニメーションさせます。
transitionの基本的な記述は 10-1 のように、アニメーションさせたいプロパティ名、秒数、イージングを順に指定します。

10-1 transitionの基本的記述

```
-moz-transition: all 1.0s linear;
-webkit-transition: all 1.0s linear;
transition: all 1.0s linear;
```



11

transitionの3つ目のオプションであるイージングでは、アニメーションの変化速度を指定します。本作例で指定しているease-outは、アニメーションが終わりに近づくにつれてゆっくりになる効果です。
イージングの値としてあらかじめ動きが決められているキーワードには、11-1のようなものがあります。

11-1 指定できるイージングキーワード

キーワード	効果
linear	一定の速度
ease	滑らかな変化
ease-in	ゆっくりと開始し、素早く終了
ease-out	素早く開始し、ゆっくりと終了
ease-in-out	ゆっくりと開始し、素早く変化し、ゆっくり終了

12

サムネイルのアニメーションはtransformをターゲットとし、0.2秒でイージングにease-outを指定します。コードは12-1のようになります。
アニメーション対象のプロパティを指定する際にもベンダープレフィックスが必要になるので、注意が必要です。
以上で、完成となります12-2。

12-1 最終的な #Gallery li

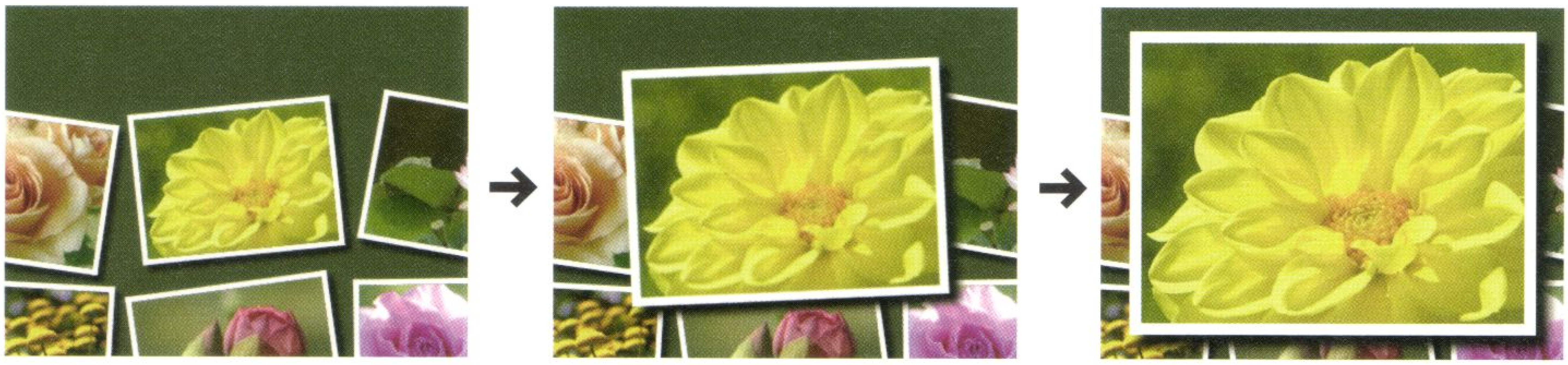
```
#Gallery li {
  float:left;
  width:160px;
  height:120px;
  margin:0 20px 20px 0;
  border:5px solid #ffffff;

  position:relative;
  z-index:10;

  -moz-box-shadow:5px 5px 5px #000000;
  -webkit-box-shadow:5px 5px 5px #000000;
  box-shadow:5px 5px 5px #000000;

  -moz-transition:-moz-transform 0.2s ease-out;
  -webkit-transition:-webkit-transform 0.2s ease-out;
  transition:transform 0.2s ease-out;
}
```

12-2 アニメーション

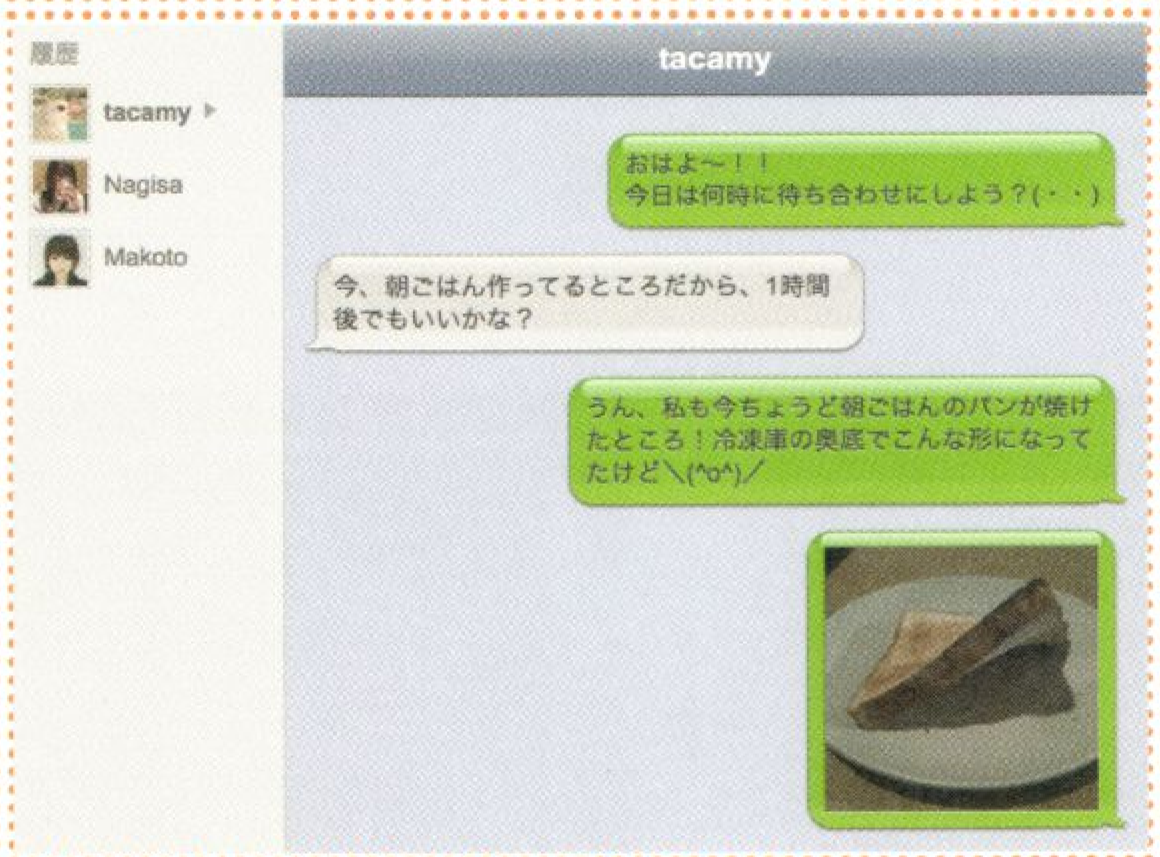


まとめ [CONCLUSION]

ロールオーバーとロールアウトなど、状態の遷移に関するアニメーションは、以上のようにスタイルシートだけでも実装することができます。サムネイルの数が増減しても、アニメーションを設定し直す必要がないところもメリットとなります。
しかし、すべての環境では動作しませんので、プログレッシブ・エンハンスメントをうまく設計したうえで活用するのがよいでしょう。

02 border-imageを活用したポラロイド風のフレーム





これまでは、borderのスタイルといえば実線や点線、破線や二重線などがメインでした。しかし、CSS3のborder-imageを使えば、borderに画像を使えるようになります。複雑すぎてCSSでは実現できなかったようなデザインを、border-imageを上手に使って実装してみましょう。

制作・文／山田敬美

制作のポイント

- 「border」という言葉にとらわれず、自由な発想で使ってみる
- 4つのborder-imageプロパティを1つずつ理解する
- ほかのプロパティとの組み合わせで、リッチなデザインに対応

対応ブラウザ

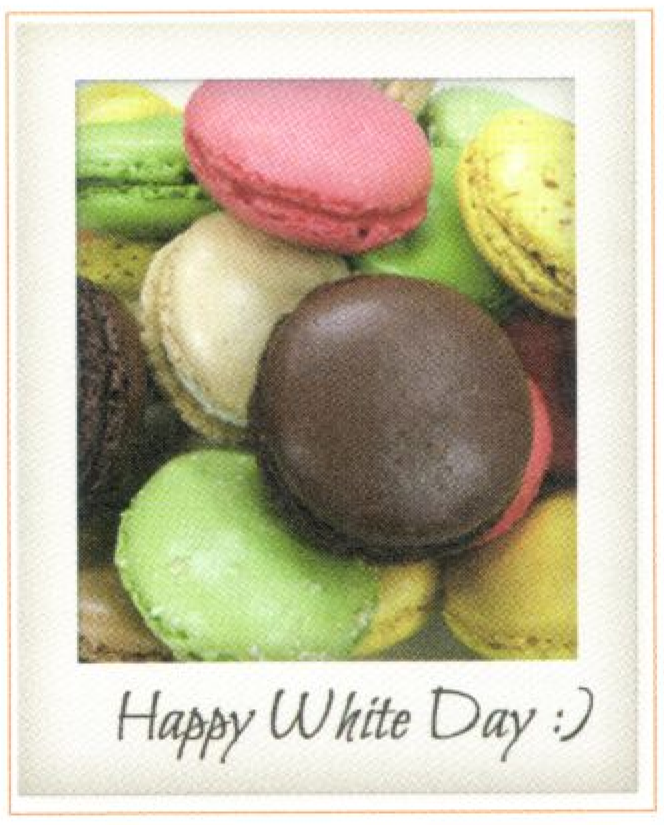
IE	非対応
Safari	3 以上
Firefox	3.5 以上
Chrome	1 以上

※IE10 Platform Previewでも非対応。
※IEでは表示内容のアクセシビリティは確保されます。

TECHNIC 1

Webフォントとの組み合わせでポラロイド風に

border-imageでフレームを、Webフォントで手書き文字を再現して、本物のポラロイド写真のようなフレームを作ることができます。



TECHNIC 2

複雑なデザインも1つの要素で簡単に実装

複雑な形のシャドウや光沢など、再現が難しいデザインも画像1枚で実装できます。





border-imageの概要

01

縦横可変サイズの画像フレームを作る場合、これまでは、background-imageを使って背景画像をいくつも組み合わせる必要がありましたが、border-imageなら1つの要素でシンプルに実装できるようになります 01-1 。

01-1 画像フレームを実装する方法の比較

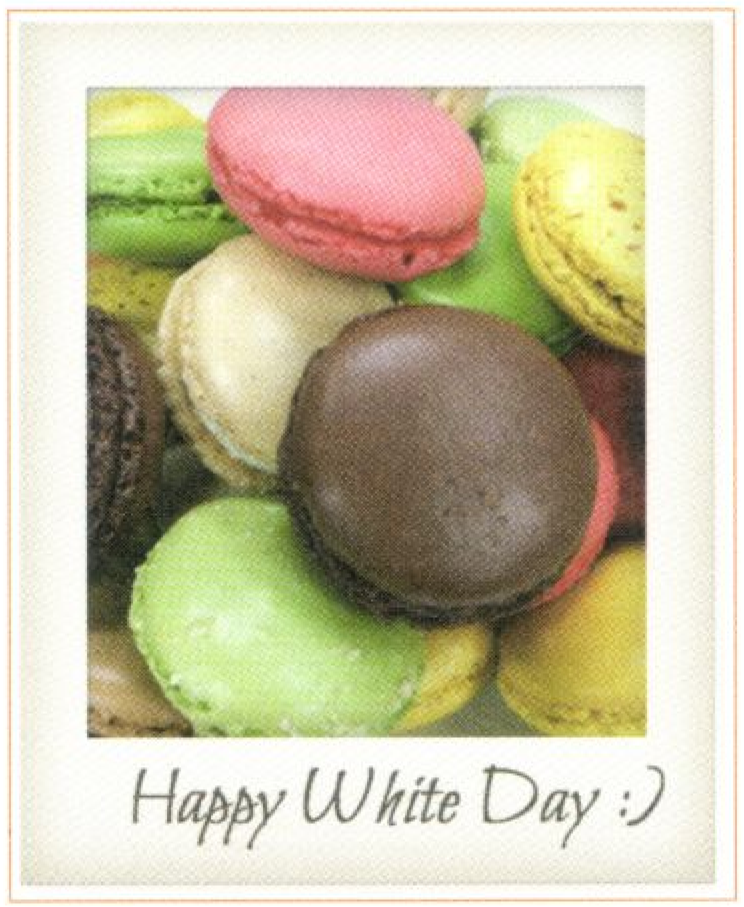
比較内容	background-image	border-image
縦横に伸縮するボックスに、画像フレームを実装する	要素を何重にも入れ子にする必要あり	必要な要素は1つのみ
ボックスサイズが、画像の繰り返しサイズの倍数でない場合	画像が途中で切れる	ボックスサイズに合わせて、ブラウザ側で画像サイズを調整するため、画像が途中で切れない（border-image-repeatがroundの場合）
Webページの印刷	デフォルト設定では印刷されない	borderと同じように印刷される

ポラロイド風の画像フレームをつくる

02

border-imageを使い、写真の周囲をポラロイド風のフレームで装飾します 02-1 。基本となるHTMLは 02-2 の通りで、無駄な要素がないシンプルな構造となっています。

02-1 完成例



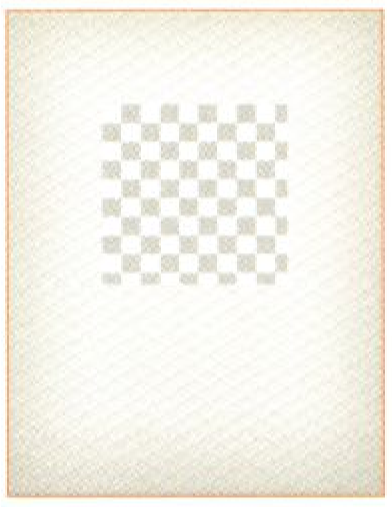
02-2 ベースのHTML

```
<figure>
  <img src="macaron.jpg">
  <figcaption>Happy White Day :)</figcaption>
</figure>
```

03

まず、フレームになる画像URLを指定します（border-image-source） 03-1 03-2 。border-imageを実装する際は、 03-3 のように画像を9分割し、それぞれの区画に対して処理をするというイメージを持つとわかりやすくなります。

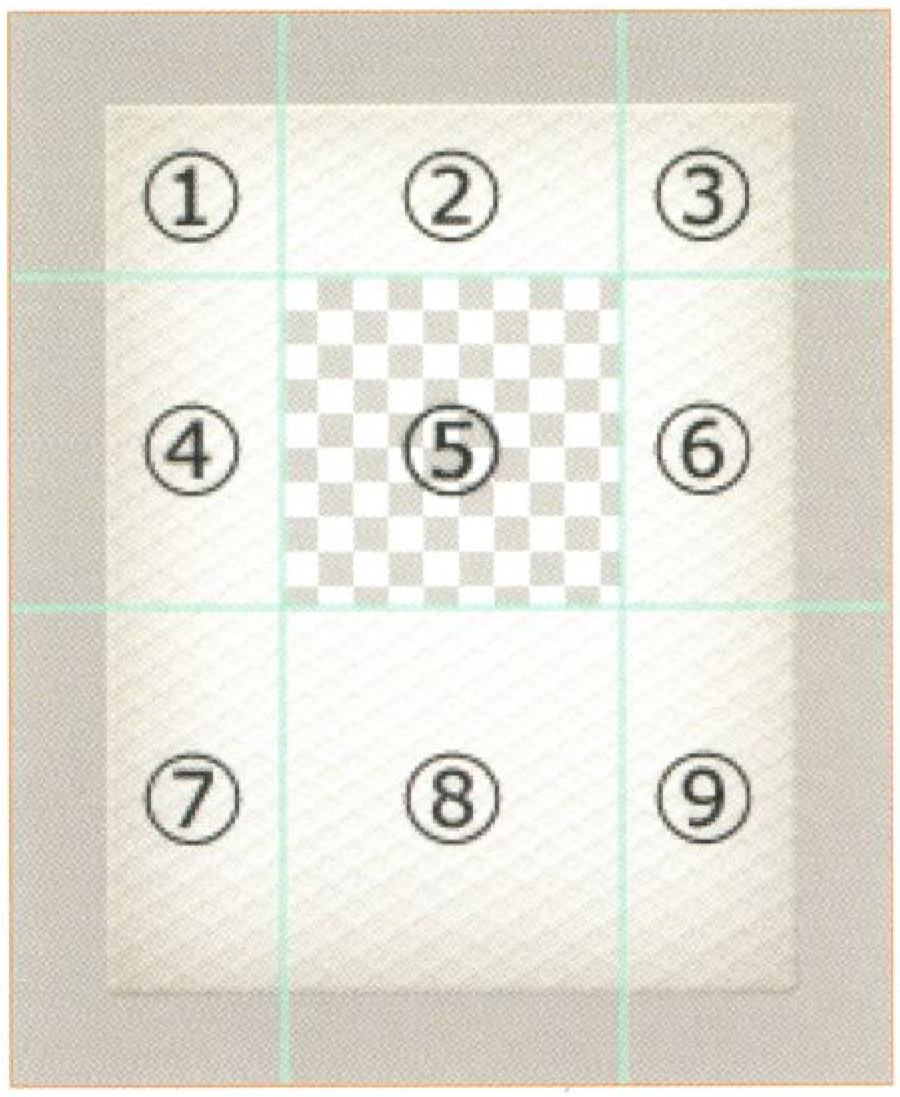
03-1 フレームになる画像



03-2 border-image-sourceの値を指定

```
figure {
  display: inline-block;
  -webkit-border-image: url(frame.png);
  -moz-border-image: url(frame.png);
  -o-border-image: url(frame.png);
  border-image: url(frame.png);
}
```

03-3 9分割のイメージ

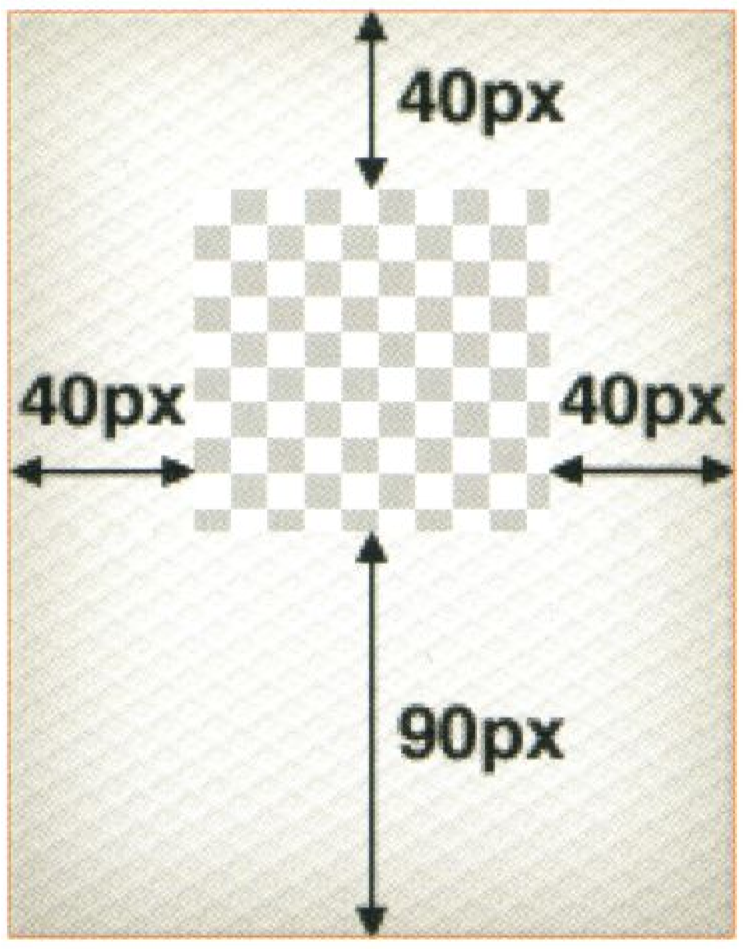


> **MEMO**
>
> 図 03-3 の、①③⑦⑨部分は画像が固定され、②④⑥⑧部分は繰り返す（または引き伸ばす）部分です。今回は繰り返しにするため、②④⑥⑧がそれぞれリピートできるような画像を作成しておく必要があります。

04

ブラウザは、1枚の画像からどのように区画を分ければよいか判断できません。そのため、上下左右の枠になる部分のサイズをCSSで指定する必要があります（border-image-slice）04-1。指定方法は、「上、右、下、左」の順に、数値を単位をつけずに記述します04-2。ただし、サイズが同じ場合は、「上、左右、下」「上下、左右」「すべて」のように短縮して記述することも可能です。

04-1 枠になる部分のサイズを測定



TIPS

「border-image-source:url(frame.png); border-image-slice:40 40 90 40;」のように、独立したプロパティで記述する方法もありますが、ほとんどのブラウザがこの記述方法に対応していないため、ここではすべてborder-imageショートハンドでまとめて指定しています（これ以降のプロパティも同様）。

ATTENTION!

Operaは短縮形式に対応していないため、「上、右、下、左」の順でそれぞれ個別に記述する必要があります。

04-2 border-image-sliceの値を指定

```
figure {
  display: inline-block;
  -webkit-border-image: url(frame.png) 40 40 90;
  -moz-border-image: url(frame.png) 40 40 90;
  -o-border-image: url(frame.png) 40 40 90 40;
  border-image: url(frame.png) 40 40 90;
}
```

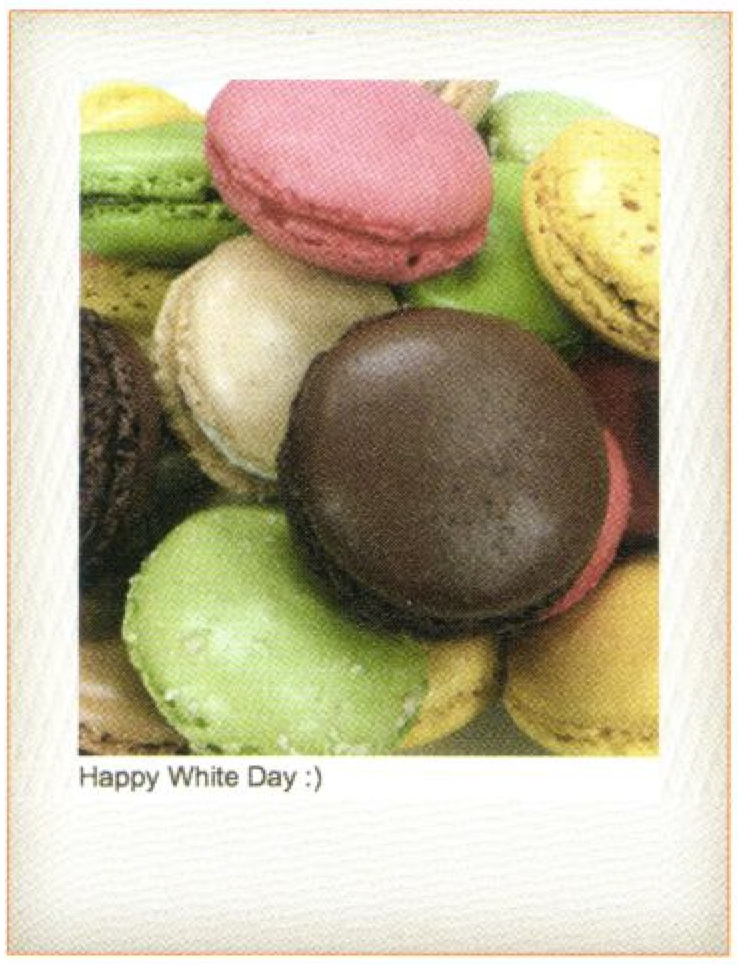
05

borderの太さを指定します（border-image-width）。border-image-sliceの値の後にスラッシュ（/）を挟み、「上、右、下、左」の順に記述します 05-1 05-2。border-image-sliceでは単位は不要でしたが、border-image-widthでは単位が必要です。また、スライスサイズと同様に、「上、左右、下」、「上下、左右」、「すべて」のように短縮することも可能です。

05-1 border-image-widthの値を指定

```
figure {
  display: inline-block;
  -webkit-border-image: url(frame.png) 40 40 90 / 40px 40px 90px;
  -moz-border-image: url(frame.png) 40 40 90 / 40px 40px 90px;
  -o-border-image: url(frame.png) 40 40 90 40 / 40px 40px 90px 40px;
  border-image: url(frame.png) 40 40 90 / 40px 40px 90px;
}
```

05-2 ②④⑥⑧の繰り返し部分が引き伸ばされた状態となる



MEMO

border-image-widthの値はpx以外にも、パーセンテージや倍数、「auto」キーワードも用意されていますが、ブラウザの対応状況がまちまちであるため、pxで指定しておく方法が無難です。



06

border-imageは、03-3の②④⑥⑧部分を繰り返すことにより、縦横可変サイズに対応できる仕組みとなっています。それをどのように繰り返すかを指定するのが、border-image-repeatです。

指定できるキーワードは「stretch」、「repeat」、「round」、「space」の4種類あり、それぞれ06-1のような仕様となっています。今回は、角の継ぎ目をわからなくさせたいため、「round」を指定します06-2。

MEMO

「stretch round」のように、「上下、左右」の2つに分けて指定することもできます。

ATTENTION!

ChromeとSafariでは「round」に対応していないため、「repeat」と同じ表示になり、角の継ぎ目が目立っています06-3 06-4。

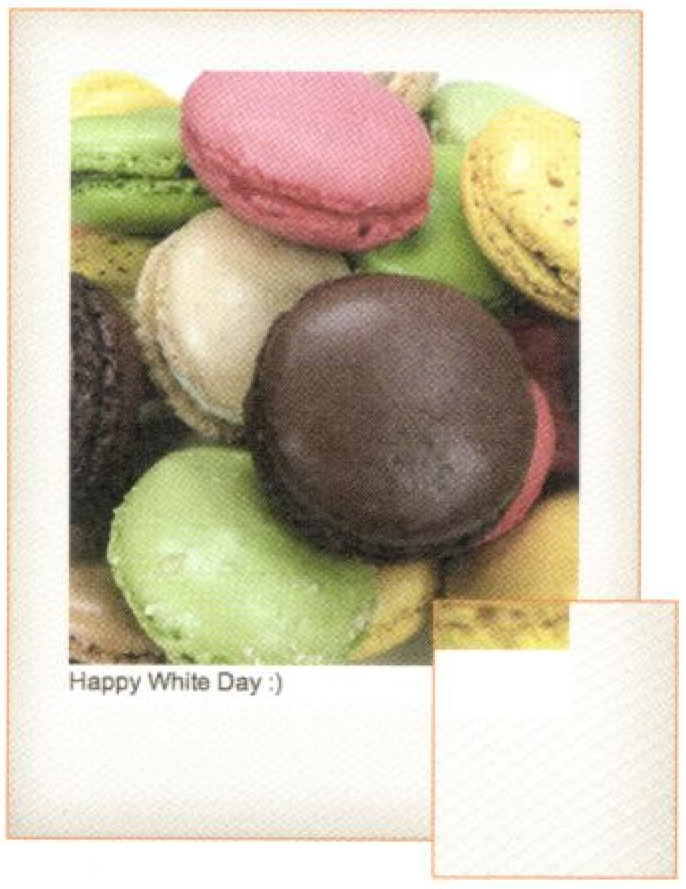
06-1 border-imageで指定できるキーワード

キーワード	説明
stretch	画像が引き伸ばされる。初期値。
repeat	画像が繰り返される。ボックスサイズが②④⑥⑧それぞれの倍数でない場合、画像は途中で切れる。
round	画像が繰り返される。ボックスサイズが②④⑥⑧それぞれの倍数でない場合、うまく収まるように画像が拡大・縮小される。
space	画像が繰り返される。ボックスサイズが②④⑥⑧それぞれの倍数でない場合、余ったスペースが画像の周囲に分配される。

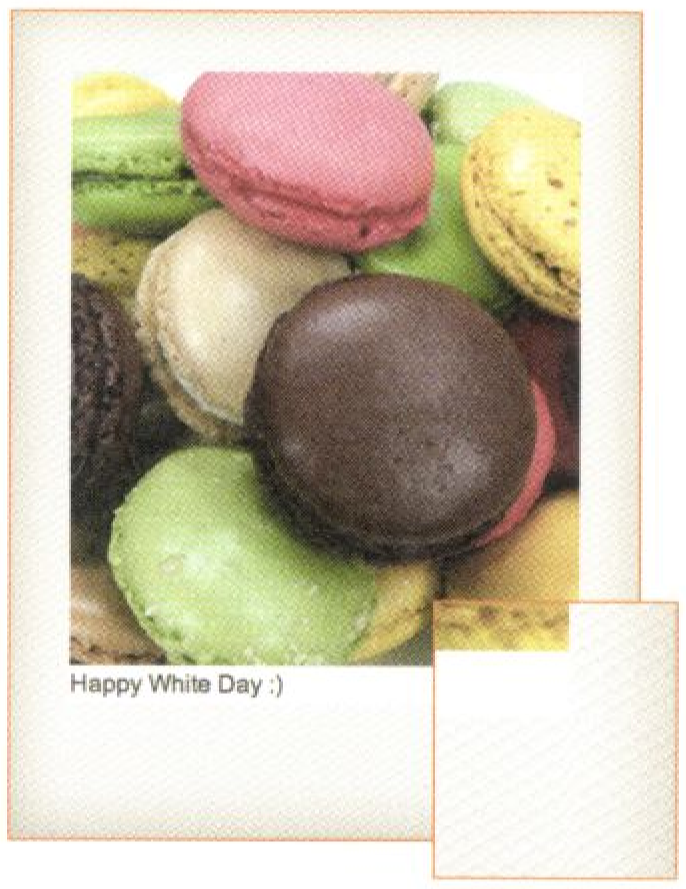
06-2 border-image-repeatの値（キーワード）を指定

```
figure {
  display: inline-block;
  -webkit-border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
  -moz-border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
  -o-border-image: url(frame.png) 40 40 90 40 / 40px 40px 90px 40px round;
  border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
}
```

06-3 Firefoxでの表示。四隅がきれいにつながっている



06-4 Chromeでの表示。四隅の継ぎ目がつながっていない



手書き風のキャプションをつける

07

border-imageに関する指定は以上です。ここからは、キャプションに対する指定を行います。

まず、Google Web Fonts 07-1を利用し、キャプションのテキストを手書き風フォントに変更します07-2 07-3 07-4。

07-1 Google Web Fonts

http://www.google.com/webfonts/

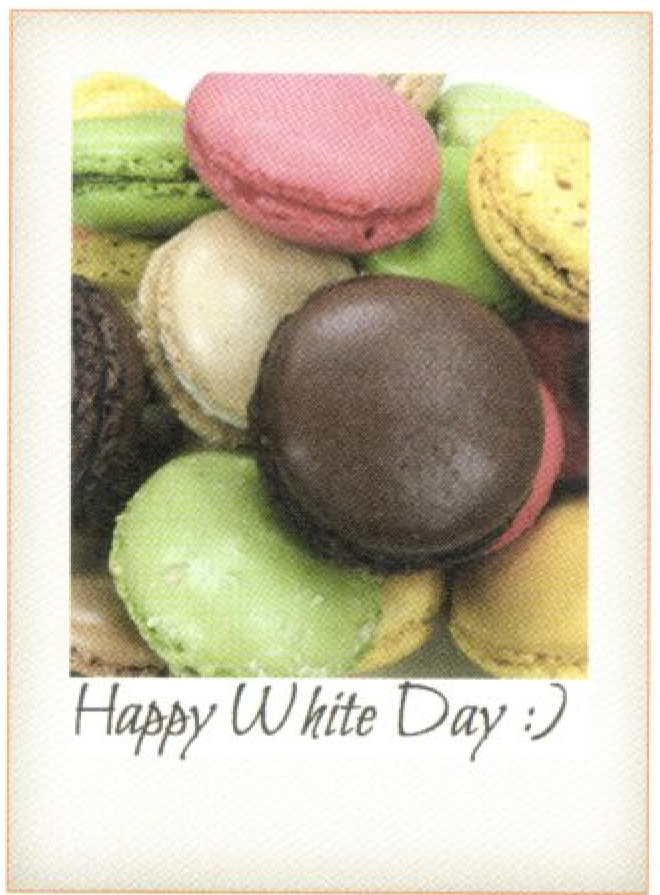
07-2 HTMLの記述

```
<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Bilbo">
```

07-3 CSSの記述

```
figure figcaption {
  line-height: 1;
  color: #333;
  font-size: 50px;
  font-family: 'Bilbo', cursive;
  white-space: nowrap;
}
```

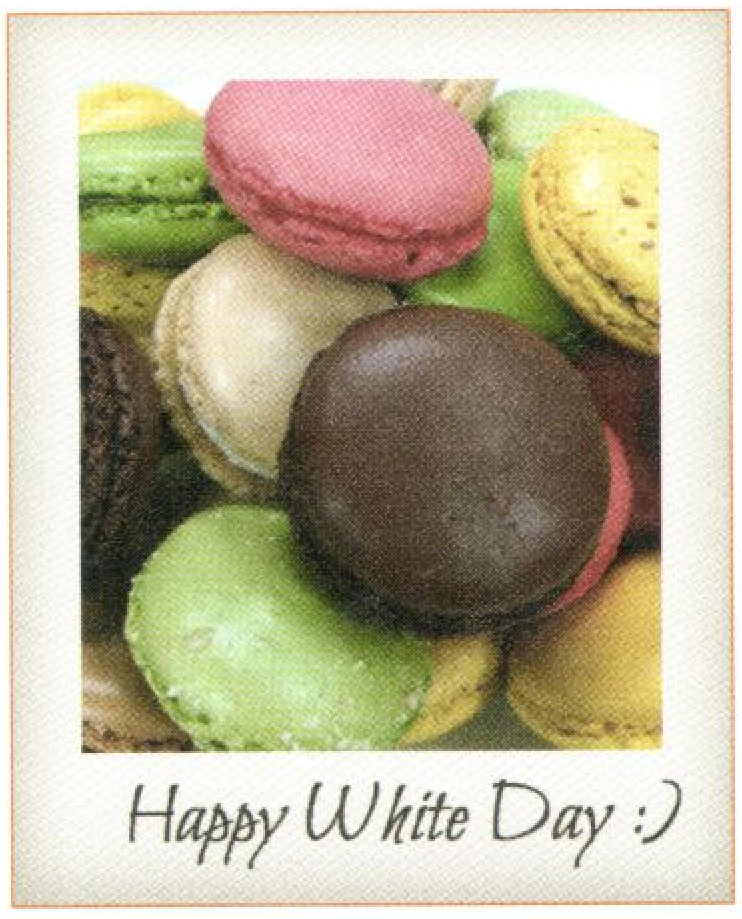
07-4 キャプションにWebフォントが適用された



08

次に、positionを使い、フレームの右下にテキストを表示させます。
figure要素にposition:relative;を指定することで、figcaptionを位置指定する際の基準要素とします。基準点はborderの外側でなく内側となるため、figcaptionの位置指定にマイナスの値を指定することで、border-imageの上に文字を乗せられます08-1 08-2。

08-2 border-imageの上にテキストが表示されるよう調整



08-1 写真の右下を基準として、キャプションの位置を指定

```
figure {
  position: relative;
  display: inline-block;
  -webkit-border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
  -moz-border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
  -o-border-image: url(frame.png) 40 40 90 40 / 40px 40px 90px 40px round;
  border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
}
figure figcaption {
  position: absolute;
  bottom: -65px;
  right: -10px;
  z-index: 1;
  line-height: 1;
  color: #333;
  font-size: 50px;
  font-family: 'Bilbo', cursive;
  white-space: nowrap;
}
```

09

キャプションの装飾は以上です。最後に、box-shadowを使って、さらにポラロイド写真を演出します。
box-shadowはカンマ（,）で区切ることにより、複数の影を1つの要素に指定できます。ここでは、左から順に「フレームと写真の間の影」、「写真全体を薄暗くする影」、「フレーム全体に立体感を出すための影」の3種類の影を指定しました09-1。
ただし、border-imageよりも写真の方が手前になるため、insetで指定した2つの指定は、写真の後ろに隠れてしまいます。そのため、09-2の指定により、写真画像をborder-imageよりも背面に持っていきます09-3。

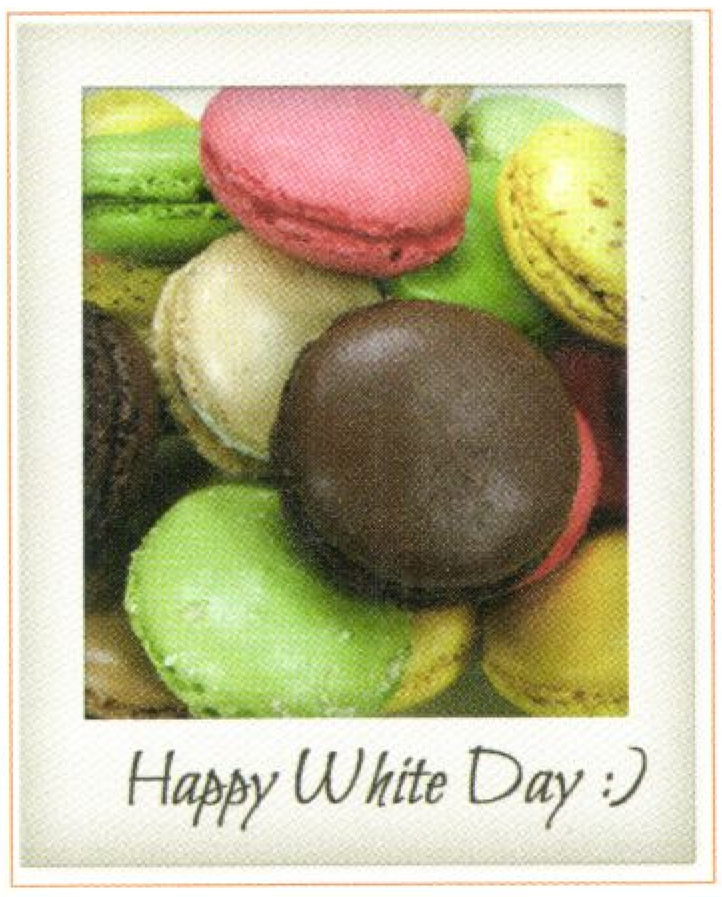
09-1 3種類のbox-shadowをカンマ区切りで指定

```
figure {
  position: relative;
  display: inline-block;
  -webkit-border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
  -moz-border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
  -o-border-image: url(frame.png) 40 40 90 40 / 40px 40px 90px 40px round;
  border-image: url(frame.png) 40 40 90 / 40px 40px 90px round;
  box-shadow: inset 2px 2px 1px rgba(0, 0, 0, 0.2), inset 0 0 30px rgba(0, 0, 0, 0.3), 3px 3px
  10px rgba(0,0,0,.1);
}
```

09-2 写真画像がborder-imageより背面になるよう指定

```
.photo img {
  position: relative;
  z-index: -1;
  vertical-align: bottom;
}
```

09-3 border-imageが写真よりも前面になることで、box-shadowが表示された





10

以上で、ポラロイド風の画像フレームが完成しました。
別サイズの画像にも、同様にフレームをつけることができます 10-1 。

10-1 ポラロイド風の画像フレームの完成



チャットアプリ風の吹き出しをつくる

11

もうひとつのバリエーションとして、凝ったデザインの吹き出しをborder-imageを使って実装します 11-1 。ベースとなるHTMLは 11-2 の通り、シンプルな構造です。

11-1 完成例



11-2 ベースのHTML

```
<!--画像-->
<div class="balloon">
  <p><img src="bread.jpg"></p>
</div>

<!--文字列-->
<div class="balloon">
  <p>テキスト</p>
</div>
```

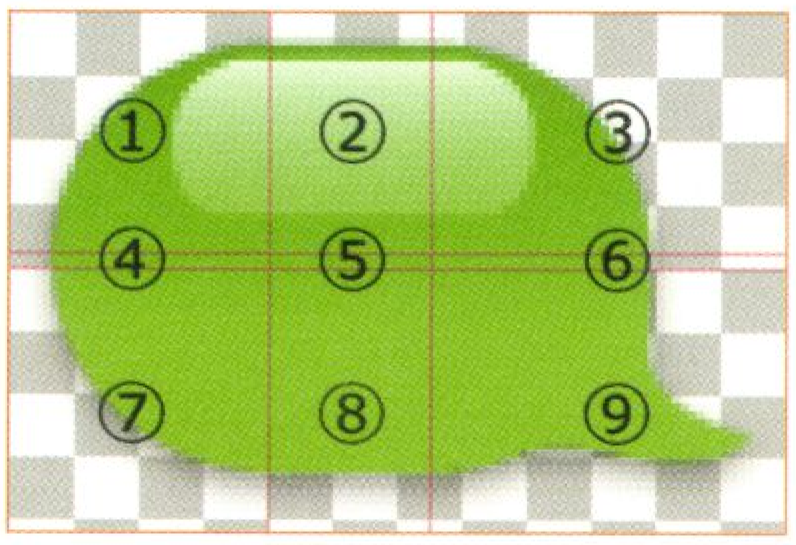
12

ポラロイド風フレームのときと同様に、画像URL（border-image-source）を指定します 12-1 。画像を9分割するイメージで、②④⑥⑧の部分をリピートできるような画像を作成しておきます 12-2 。

12-1 border-image-sourceの指定

```
.balloon {
  display: inline-block;
  max-width: 60%;
  -webkit-border-image: url(balloon01.png);
  -moz-border-image: url(balloon01.png);
  -o-border-image: url(balloon01.png);
  border-image: url(balloon01.png);
}
```

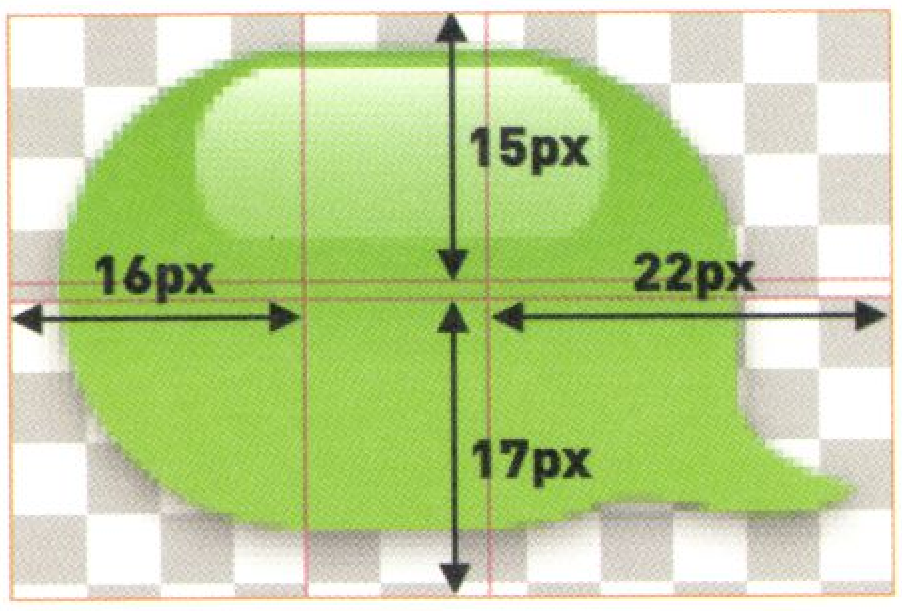
12-2 balloon01.png



13

スライスサイズ（border-image-slice）とborderの太さ（border-image-width）を指定します。13-1の図の通り、すべての辺のサイズが異なるため、13-2のように「上、右、下、左」の順で個別に指定します。

13-1 balloon01.pngのスライスサイズ



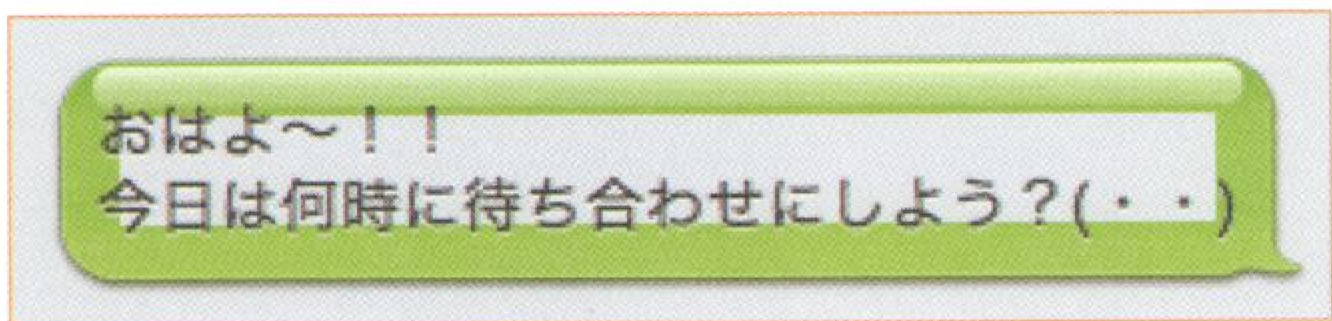
13-2 border-image-sliceとborder-image-widthの指定

```
.balloon {
  display: inline-block;
  max-width: 60%;
  -webkit-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px;
  -moz-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px;
  -o-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px;
  border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px;
}
```

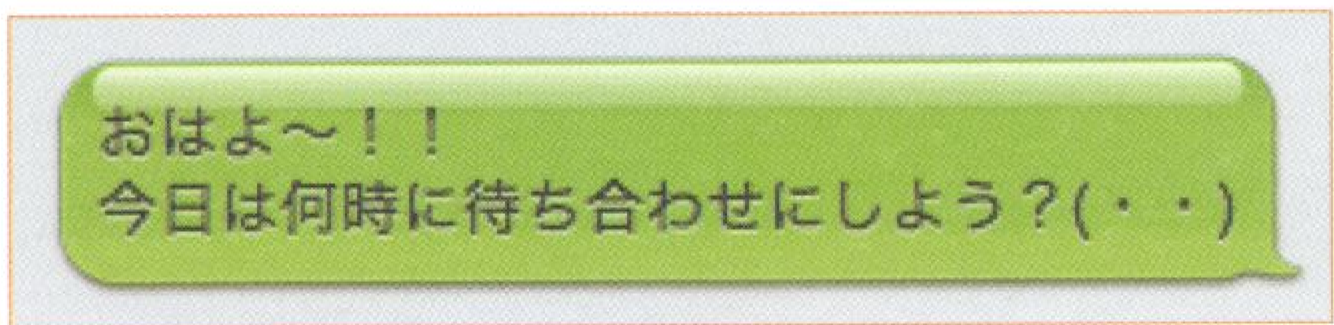
14

Chromeで表示すると、中央部分の画像が抜けて透明になっています14-1。border-imageの中央部分（12-2の⑤の部分）の画像は、デフォルトでは使われない仕様であるため、中央の画像も表示したい場合は、border-image-sliceの数値に続けて「fill」キーワードを指定します14-2 14-3。

14-1 Chromeでの表示。中央部分が抜けている



14-3 中央部分の画像も表示された



14-2 fillキーワードを追加

```
.balloon {
  display: inline-block;
  max-width: 60%;
  -webkit-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px
  17px 16px;
  -moz-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px
  17px 16px;
  -o-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px;
  border-image: url(balloon01.png) 15 22 17 16 fill / 15px 22px 17px 16px;
}
```

*COLUMN

fillキーワードに注意

fillキーワードは後から追加された仕様です。それ以前は、デフォルトの状態で中央部分の画像も表示されていました。そのため、ベンダープレフィックスつきのプロパティにfillキーワードを指定すると、それを認識できずにborder-image自体が表示されなくなってしまいます。
fillキーワードは、ベンダープレフィックスなしのプロパティだけにつけるようにします。なお、最新のChromeはベンダープレフィックスなしのborder-imageの指定を読みにいきます。



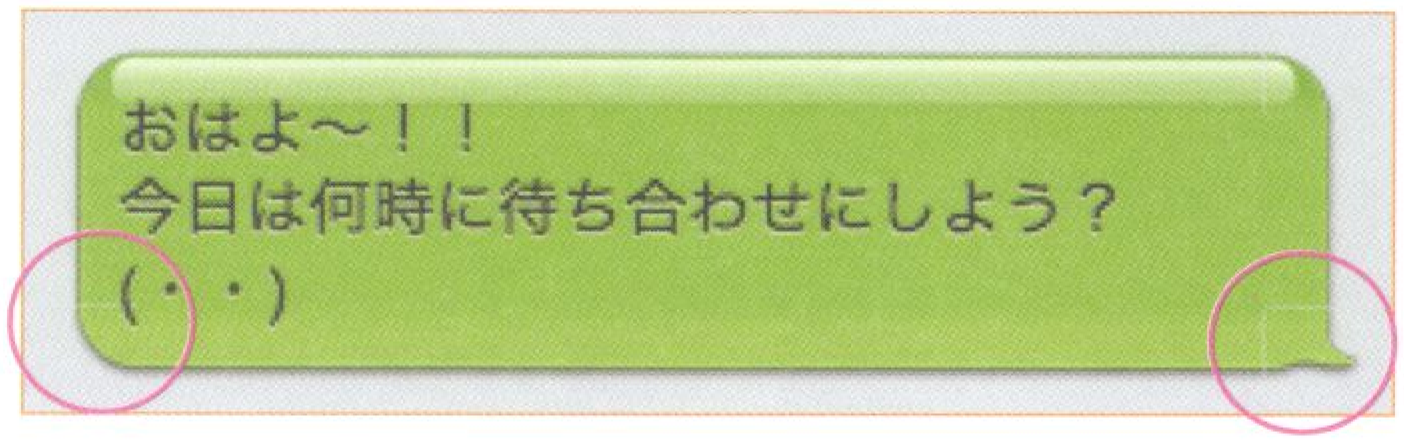
15

リピート方法を指定します（border-image-repeat）。理論的には「strech」や「repeat」でも問題ないのですが、Firefoxで1px以下の隙間ができる場合があります 15-1。「round」にすると解消されることがあるため、ここでは「round」にしています 15-2 15-3。

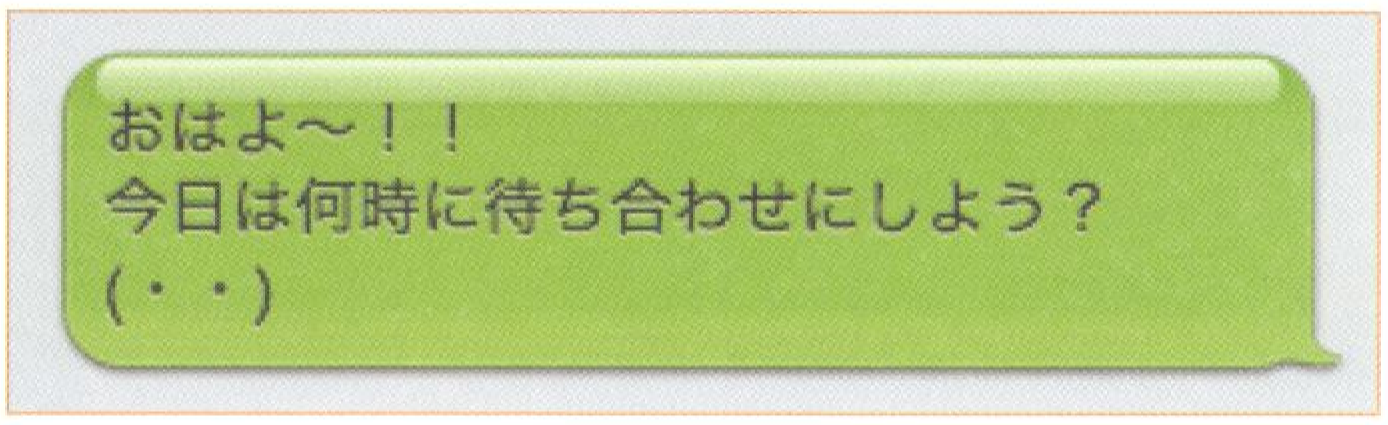
15-2 border-image-repeatをroundに指定

```
.balloon {
  display: inline-block;
  max-width: 60%;
  -webkit-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px round;
  -moz-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px round;
  -o-border-image: url(balloon01.png) 15 22 17 16 / 15px 22px 17px 16px round;
  border-image: url(balloon01.png) 15 22 17 16 fill / 15px 22px 17px 16px round;
}
```

15-1 Firefoxでの表示。右下と左下に隙間が空いている



15-3 Firefoxでの隙間が消えた



16

以上で、チャットアプリ風の吹き出しの完成です。中にどのような要素が入っても、自由に伸縮するリッチなデザインの吹き出しになりました 16-1。

16-1 チャットアプリ風の吹き出しの完成

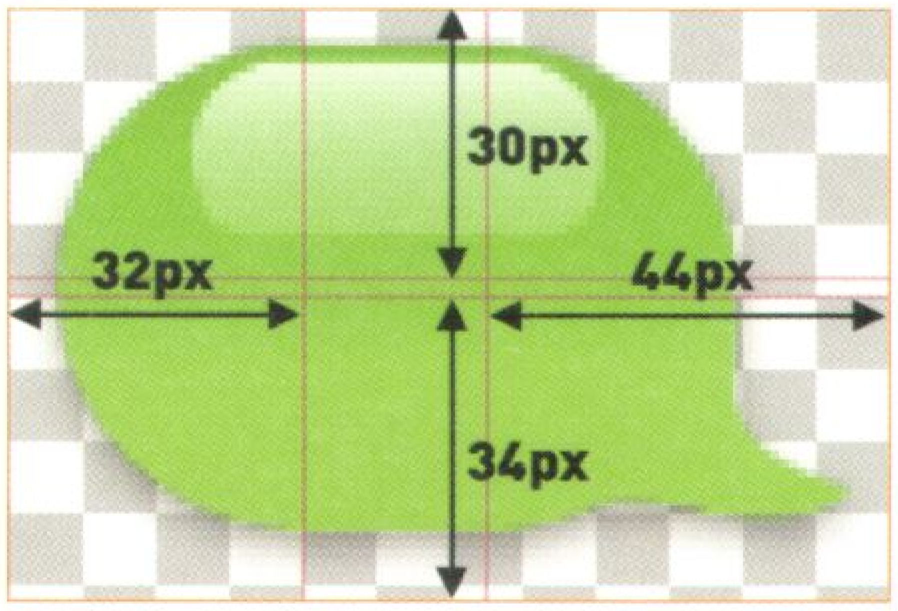


border-imageをRetinaディスプレイに対応させる

17

最近のスマートフォンやタブレット端末は、ピクセル密度（device-pixel-ratio）が1.5や2であることが多く、等倍サイズの画像をそのまま使うと滲みやジャギーで表示が汚くなります。背景画像やimg要素の場合には、2倍サイズの画像を用意しておき、background-sizeやwidthで半分に縮小して表示するのですが、border-imageの場合は2倍サイズの画像を用意し 17-1、border-image-widthの値をその半分にすることで対応できます 17-2。

17-1 2倍サイズの画像を用意



17-2 border-image-sliceは2倍のサイズで指定し、border-image-widthはその半分（表示サイズ）を指定する

```
.balloon {
  display: inline-block;
  max-width: 60%;
  -webkit-border-image: url(../img/balloon01.png) 30 44 34 32 / 15px 22px 17px 16px round;
  -moz-border-image: url(../img/balloon01.png) 30 44 34 32 / 15px 22px 17px 16px round;
  -o-border-image: url(../img/balloon01.png) 30 44 34 32 / 15px 22px 17px 16px round;
  border-image: url(../img/balloon01.png) 30 44 34 32 fill / 15px 22px 17px 16px round;
}
```

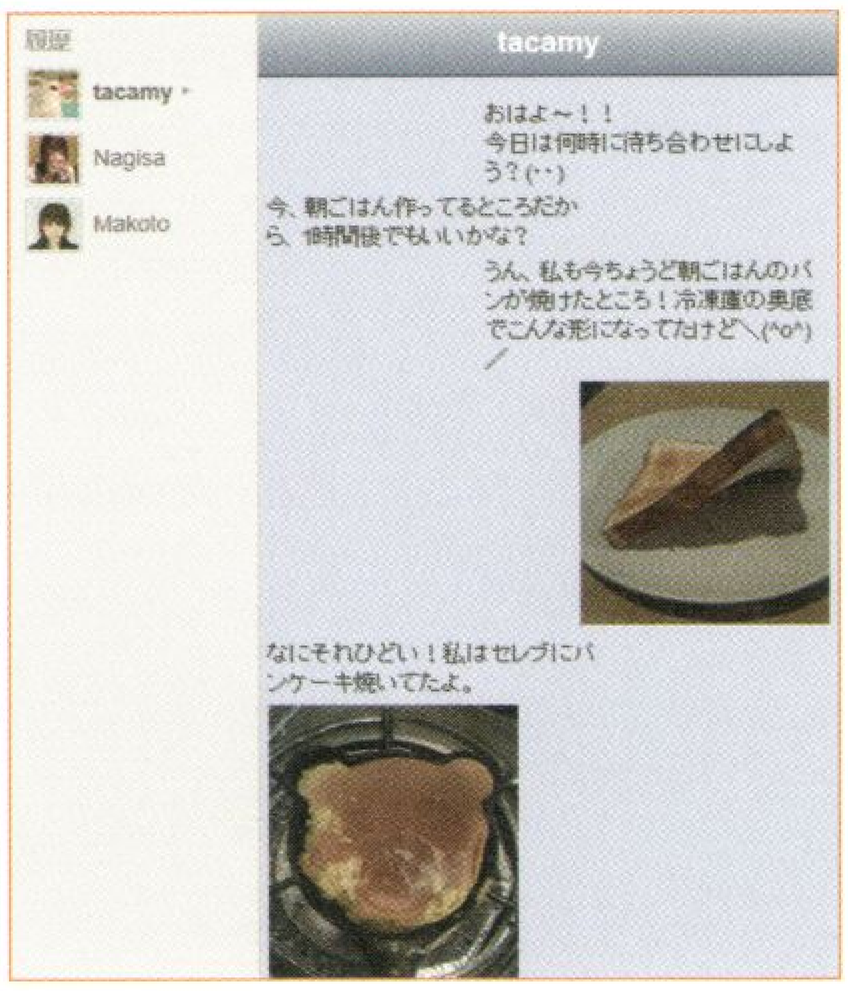
Internet Explorerへの対応状況

18

このようにとても便利なborder-imageですが、残念ながら現行のIEでは非対応となっており、さらにIE10 Platform Previewでも対応していません 18-1。IEをCSS3に対応させるJavaScriptライブラリを使ったとしても、まったく同じように表示するのは困難です。

JavaScriptライブラリ「CSS3 PIE」 18-2 ではborder-imageも対応していますが、今回のデモでは動きませんでした。

18-1 IE9での表示。border-imageの指定がまったく適用されていない



18-2 CSS3 PIE

http://css3pie.com/

MEMO

CSS3 PIEのドキュメントに、border-image-repeat:stretch;の場合でしか使えない旨の記載があります。

ATTENTION!

IEに対応するためにJavaScriptライブラリ「flexie.js」（http://flexiejs.com/）を使用しています。
flexie.jsは、ローカル環境やDropboxのPublicなどでは動作しません。Apache等でローカルにサーバーを立てるか、実際にサーバーにアップロードして動作を確認する必要があります。

19

そこで、border-imageに未対応のブラウザ向けに、borderも同時に指定しておくとよいでしょう 19-1 19-2。まったく同じ見た目にはならなくても、そのエリアがほかと区別されていることは伝えることができます。

19-2 borderも同時に指定する

```
.balloon {
  /* for IE */
  border: 15px solid #8bd21c;
  background-color: #8bd21c¥9;

  /* for Modern Browser */
  -webkit-border-image: url(../img/balloon01.png) 30 44 34 32 / 15px 22px 17px 16px round;
  -moz-border-image: url(../img/balloon01.png) 30 44 34 32 / 15px 22px 17px 16px round;
  -o-border-image: url(../img/balloon01.png) 30 44 34 32 / 15px 22px 17px 16px round;
  border-image: url(../img/balloon01.png) 30 44 34 32 fill / 15px 22px 17px 16px round;
}
```

19-1 IE向けにborderとbackground-colorを指定



まとめ [CONCLUSION]

border-imageはいまいち使いどころがわからないという声もあるようですが、縦横可変サイズに簡単に対応でき、さらにブラウザ側で継ぎ目がわからないように自動調整してくれるという、とても便利な機能が詰まったプロパティです。
「border」という固定概念にとらわれることなく、今回の吹き出しのように身近な要素に応用すれば、さまざまな場面で活用できるでしょう。



*COLUMN

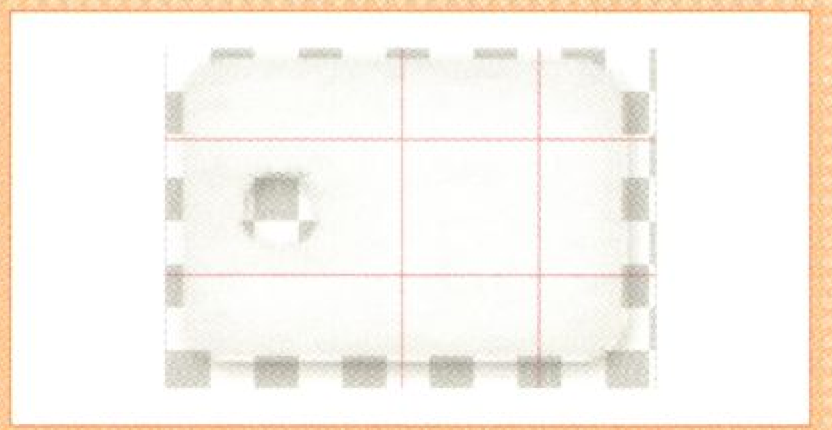
border-imageを使ったデザインアイデア

画像フレーム以外での、border-imageの使いどころをいくつか挙げてみましょう。
ルーズリーフのようなデザインで高さが可変の場合、これまでの背景画像を繰り返すという方法では、高さによっては穴が欠けてしまい、見栄えがよくありませんでした。しかし、01のようにborder-image-repeatで「round」を指定すれば、自動的に繰り返し部分のサイズを調整してくれるため、どのような高さの場合でも穴が欠けることはありません02 03。
ただし、ChromeとSafariでは「round」に対応していないため、「repeat」と同じ表示になり、高さによっては04のように穴が欠けてしまいます。

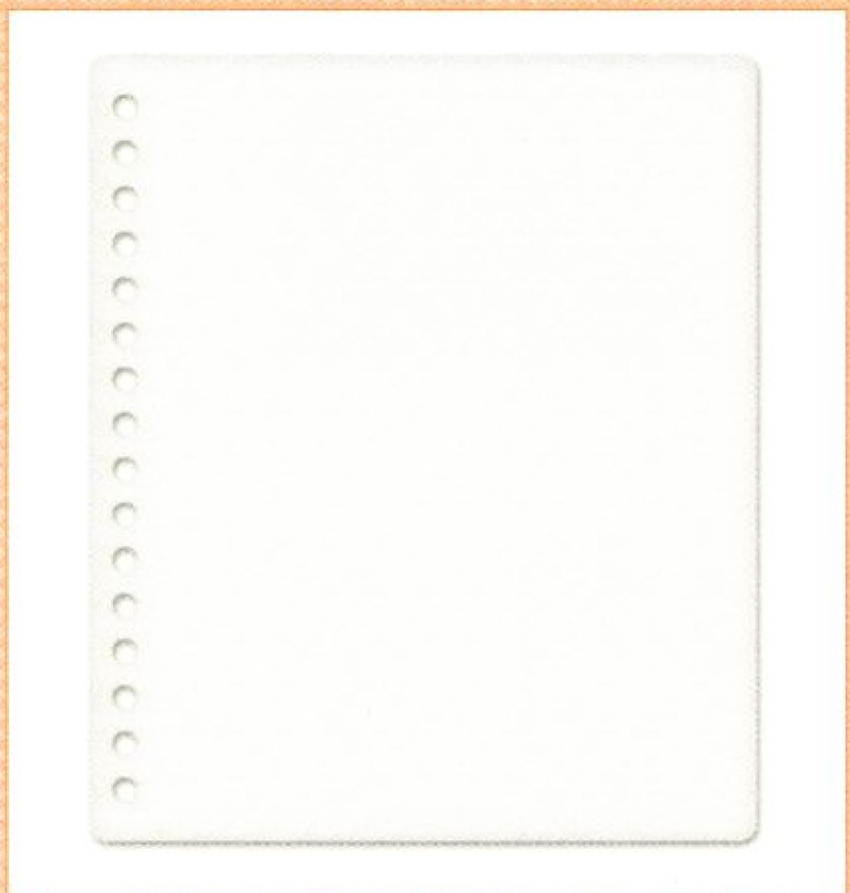
01 ルーズリーフの画像の繰り返し

```
.looseleaf {
  width: 300px;
  height: 400px;
  border: 20px solid #eee; /* for IE */
  -webkit-border-image: url(looseleaf.png) 17 21 21 43 / 17px 21px 21px 43px round;
  -moz-border-image: url(looseleaf.png) 17 21 21 43 / 17px 21px 21px 43px round;
  -o-border-image: url(looseleaf.png) 17 21 21 43 / 17px 21px 21px 43px round;
  border-image: url(looseleaf.png) 17 21 21 43 fill / 17px 21px 21px 43px round;
}
```

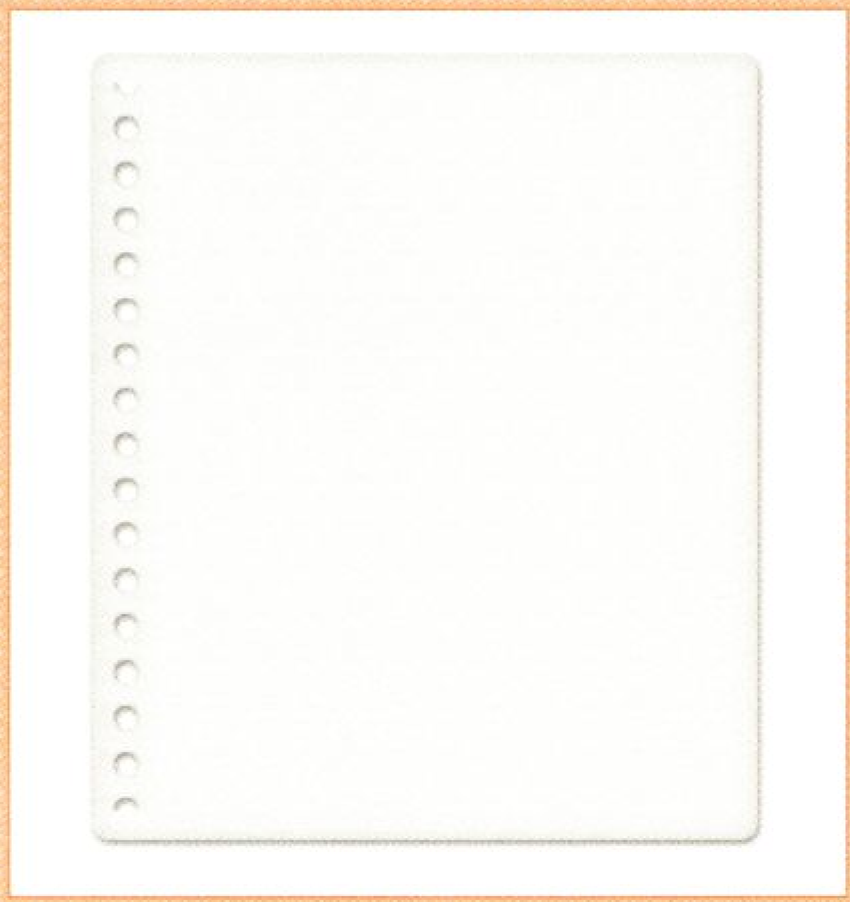
02 ルーズリーフの使用画像



03 ルーズリーフの完成イメージ(Firefoxでの表示)



04 ルーズリーフのChromeでの表示

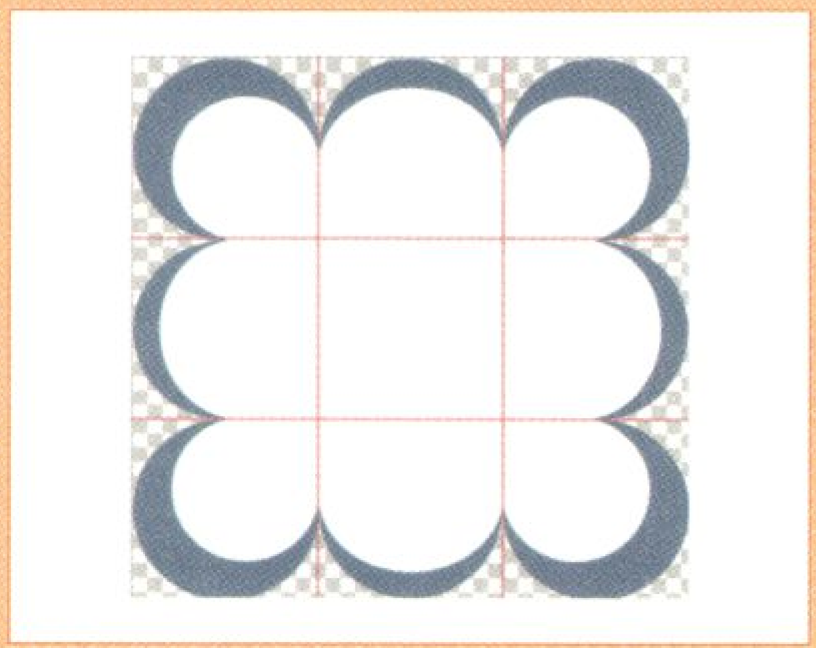


POPなどで使われそうな、雲のフレームも同様に実装できます05 06 07。ただし、これは「round」に対応していないChromeとSafariで08のように大幅に崩れるため、実際の利用はそれらに「round」が実装された後がよいでしょう。

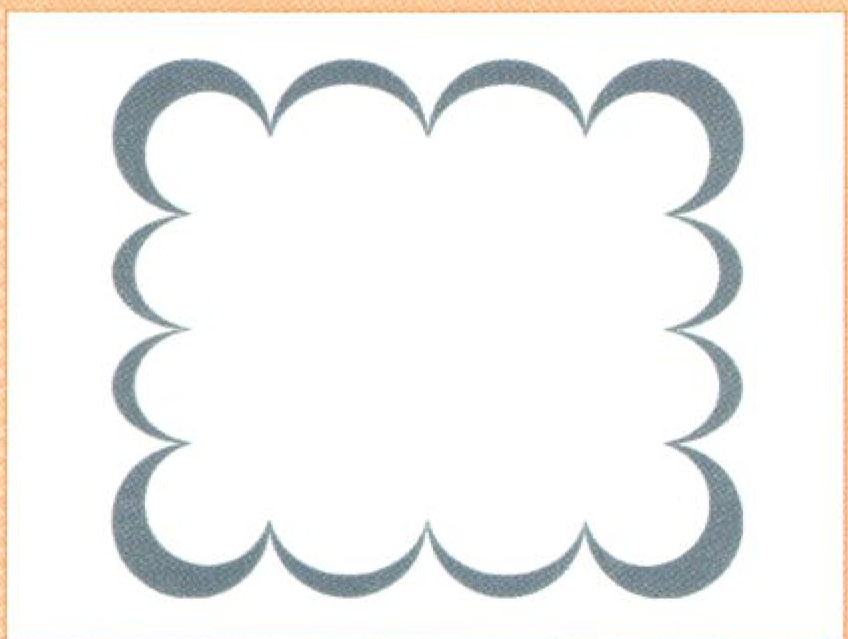
05 雲のフレームの画像の繰り返し

```
.cloud {
  width: 200px;
  height: 150px;
  border: 20px solid #2f709d; /* for IE */
  -webkit-border-image: url(cloud.png) 100 / 100px round;
  -moz-border-image: url(cloud.png) 100 / 100px round;
  -o-border-image: url(cloud.png) 100 100 100 100 / 100px 100px 100px 100px round;
  border-image: url(cloud.png) 100 fill / 100px round;
}
```

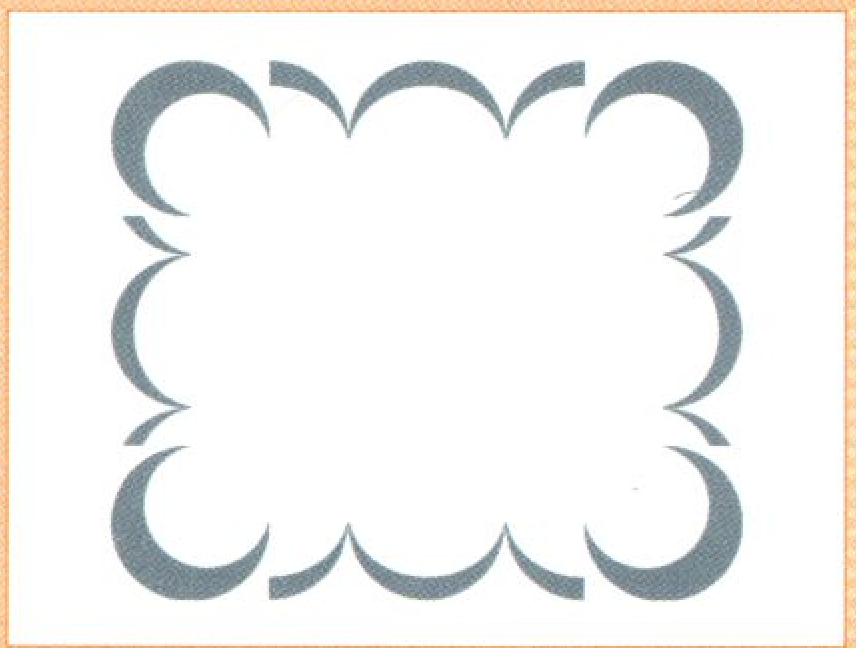
06 雲のフレームの使用画像



07 雲のフレームの完成イメージ(Firefoxでの表示)



08 雲のフレームのChromeでの表示



03 マスク効果を利用した印象的なグラフィック



↓



maskプロパティで画像にマスク効果を加えることができます。
背景画像にマスクをかけ、動きを加えることで、単なる画像だけのグラフィックではできない表現をしてみましょう。

制作・文／山本圭助
（株式会社ラナデザインアソシエイツ）

制作のポイント

- maskプロパティの使い方
- アニメーションを使ったマスクの応用
- グラデーションを使って文字列でマスクする

対応ブラウザ

IE	P.E.	Safari	5 以上
Firefox	P.E.	Chrome	15 以上

※IE、Firefoxではmaskプロパティに非対応。

TECHNIC 1

マスク効果を手軽に実現

背景画像とマスク画像を指定するだけで、マスク効果を簡単につけることができます。



TECHNIC 2

動きのあるマスクでより印象的に

背景画像を動かしたり、グラデーションをつけることで、マスクにアニメーション効果を持たせることもできます。



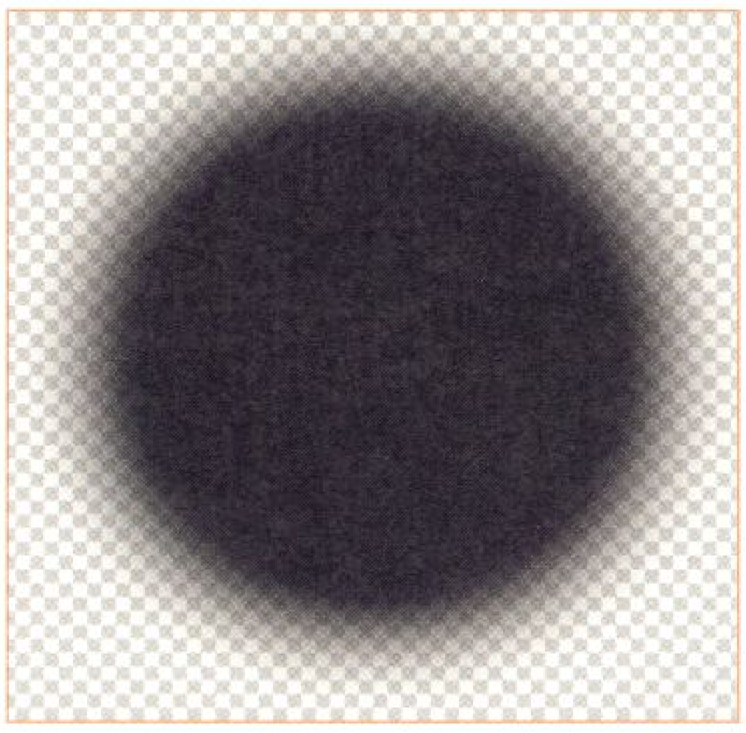


maskプロパティの使い方

01

maskプロパティは、要素の背景をマスクする画像を指定することができます。
マスクする画像 01-1 とマスクされる背景画像 01-2 を用意して、01-3 のように記述します。
すると、01-4 のように、マスクがかかった表示になります。

01-1 マスクする画像



01-4 マスク画像



01-2 背景画像



01-3 マスクの指定

```
// HTML
<body>
<div id="wrapper">
  <img src="img/bg.jpg">
</div>
</body>

// CSS
#wrapper {
  -webkit-mask-image: url(img/mask-circle.png);
}
```

*COLUMN

maskプロパティを使った効果

maskプロパティはまだChromeやSafariなどWebKit系のブラウザしか対応していませんので、WebKitのベンダープレフィックスをプロパティ名につけて使用します 01。
指定方法は、背景を指定するbackgroundプロパティに似ています。繰り返し（repeat）と配置（position）を指定できます 02。
1つにまとめて記述することもできます 03。

01 maskプロパティの指定方法

プロパティ名	機能
-webkit-mask-image:	使用する画像パスを指定
-webkit-mask-repeat	リピートする場合はrepeat、しない場合はno-repeatを指定。省略時はrepeatです
-webkit-mask-position:	右方向 下方向の順で画像の左上位置を指定

02 maskプロパティの使用例

```
-webkit-mask-image: url(mask.png);
-webkit-mask-repeat no-repeat;
-webkit-mask-position: 0 10px;
```

03 1行で記述した例

```
-webkit-mask: url(mask.png) no-repeat 0 10px;
```

02

文字の画像で背景画像をマスキングします。
ただし、maskプロパティはWebKit系ブラウザしか使えないため、非対応ブラウザではマスクする文字の画像をそのまま使用します。
まず、非対応ブラウザ用にコードを記述します 02-1 。
text-indentプロパティでテキストをとばし、backgroundプロパティで背景を表示させます 02-2 。

02-1 非対応のブラウザ向けコード

```
// HTML
<div id="container">
  <h1>good interior, good life.</h1>
</div>

// CSS
h1 {
  height: 300px;
  background: url(../img/mask-type.png) no-repeat;
  text-indent: -9999px;
}
```

02-2 非対応ブラウザでの表示



03

次に、マスクの指定をします。
WebKitだけに適用させるため、メディアクエリを使用して、ほかのブラウザを除外します 03-1 。
min-device-pixel-ratioは、画像の1pxがデバイスの上の何pxに相当するのかを特定する媒体特性です。
その最小値が0のもの、つまりすべてのデバイスを指定しています。
ベンダープレフィックスをつけることで、WebKitのデバイスが特定されます。

03-1 メディアクエリでWebKitだけを指定

```
@media screen and (-webkit-min-device-pixel-ratio: 0) {
— 省略 —
}
```

04

背景画像とマスク画像を指定します 04-1 04-2 04-3 。
2つの画像が合成され、 04-4 のようになります。

04-1 背景画像





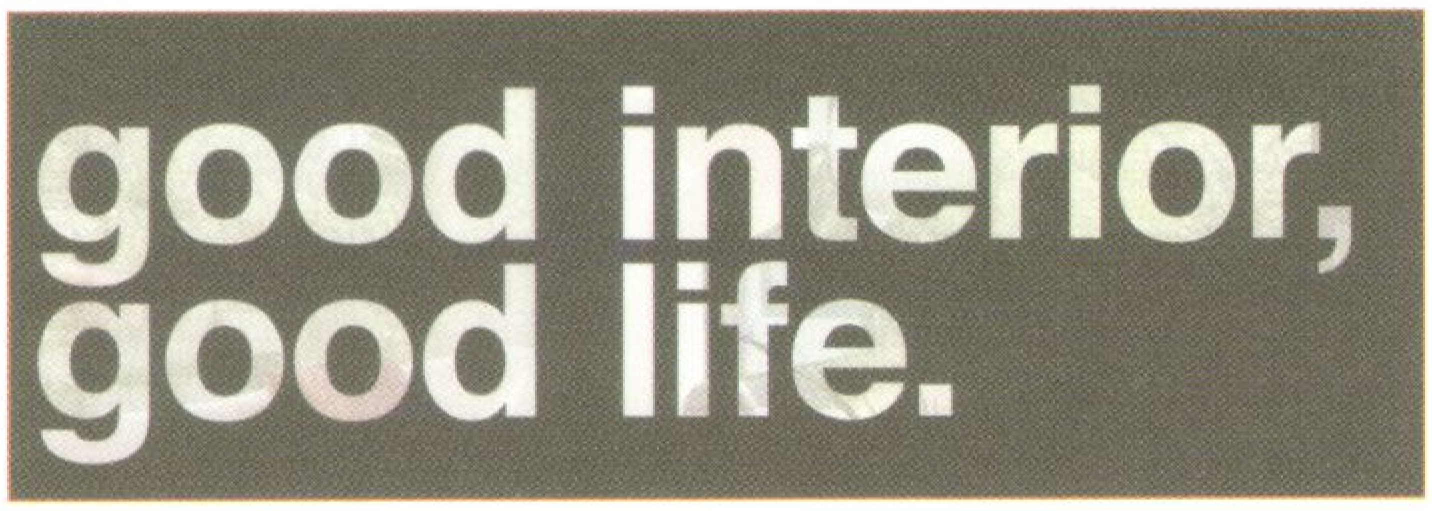
04-2 マスク画像



04-3 マスクをかける

```
@media screen and (-webkit-min-device-pixel-ratio: 0) {
  h1 {
    background: url(../img/bg-type.jpg) repeat -60px -60px;
    -webkit-mask: url(../img/mask-type.png) no-repeat 0px 0px;
  }
}
```

04-4 合成結果



アニメーションを使ったマスクの応用

05

単に画像でマスクするだけでは、画像で切り出したのと変わりません。アニメーションを追加して、マスクの特徴をより活かしてみましょう。
@keyframesを使って、アニメーションを設定します 05-1。現時点では、@keyframesやanimationプロパティには、ベンダープレフィックスが必要です。
不透明度を0から1まで変化させることで、フェード効果が出ます。ここでは、開始時にはまったく見えない状態で、経過時間が75%に達した時に完全に見えるように設定します。
また、背景画像（bg-type.jpg）の配置を開始から終了までずらすことで画像が移動するアニメーション効果も追加しておきます。

05-1 アニメーションを設定する

```
@-webkit-keyframes mask-anime {
  0% {
    opacity: 0;
    background-position: -24px -24px;
  }
  75% {
    opacity: 1;
  }
  100% {
    background-position: -60px -60px;
  }
}
```

06

@keyframesで作成したアニメーションを、animationプロパティで呼び出します 06-1。
これで、マスクで切り出した部分だけがアニメーションされるようになりました。

06-1 アニメーションを呼び出す

```
@media screen and (-webkit-min-device-pixel-ratio: 0) {
  h1 {
    -webkit-animation: mask-anime 4s;
  }
}
```

*COLUMN

CSSでアニメーション

@keyframesでアニメーションを作成し 01、animationプロパティで実行させます 02。一括で指定することもできます 03。name以外は省略時にデフォルト値が適用されます。

01 @keyframesの記述

```
@keyframes アニメーション名 {
  0% または from {
    開始。変更したいプロパティと値を記述。
  }
  100% または to {
    終了。変更したいプロパティと値を記述。
  }
}
```

02 animationプロパティ

値	効果
animation-name:	@keyframesで指定したアニメーション名
animation-duration:	1回のアニメーションにかける時間。単位はs（秒）かms（ミリ秒）でデフォルトは0s（0の時も単位が必要）
animation-timing-function:	アニメーション時のイージング。デフォルトはease
animation-delay:	開始までの待ち時間
animation-iteration-count:	実行回数。デフォルトは1。inifiniteで無限回数も指定可能
animation-direction:	方向。デフォルトはnormalで一方方向。alternateで偶数回目のアニメーションを逆転
animation-fill-mode:	アニメーション実行前後に@keyframesの0%(from)や100%(to)に指定した値を適用するかどうか

03 1行で指定した例

```
animation: name duration timing-function delay iteration-count direction fill-mode
```

*COLUMN

アニメーションできるプロパティ

数値で指定するタイプのプロパティや色指定のプロパティはアニメーションが可能です。具体的には、以下のようなプロパティがあてはまります。

bottom, top, left, right
background-color
background-position
border
color
font-size
font-weight
height, min-height, max-height
letter-spacing
line-height
margin
opacity
outline-width
outline-offset
outline-color
padding
text-indent
text-shadow
vertical-align
width, min-width, max-width
word-spacing
visibility
z-index



グラデーションを使って文字列でマスクする

07

マスクは文字列でかけることもできます。グラデーション機能とあわせて、文字列にグラデーションがかかるようにしてみましょう。
通常、文字列の色はcolorで指定するので、グラデーションをかけたり画像を指定したりすることはできません。しかし、maskプロパティを使えば、指定した画像を文字列で切り抜くことができるので、グラデーションのような表現が可能になります。
グラデーションのかかった画像を用意してもよいのですが、ここではgradientを使って、上から下へ透明になっていく画像を生成します 07-1 07-2 。

07-1 gradientの記述

```
-webkit-gradient(linear, 開始位置, 終了位置, from(開始色), to(終了色));
```

07-2 上から下へ透明になっていく画像を生成する

```
-webkit-gradient(linear, left top, left bottom, from(rgba(0, 0, 0, 1)), to(rgba(0, 0, 0, 0.2)));
```

ATTENTION!

-webkit-gradientは、-moz-linear-gradientと同様の記述法も可能です 01 。

01 -moz-linear-gradientと同様の記述例

```
-webkit-linear-gradient(方向, 開始色, 終了色);
-webkit-linear-gradient(top, rgba(0, 0, 0, 1), rgba(0, 0, 0, 0.2));
```

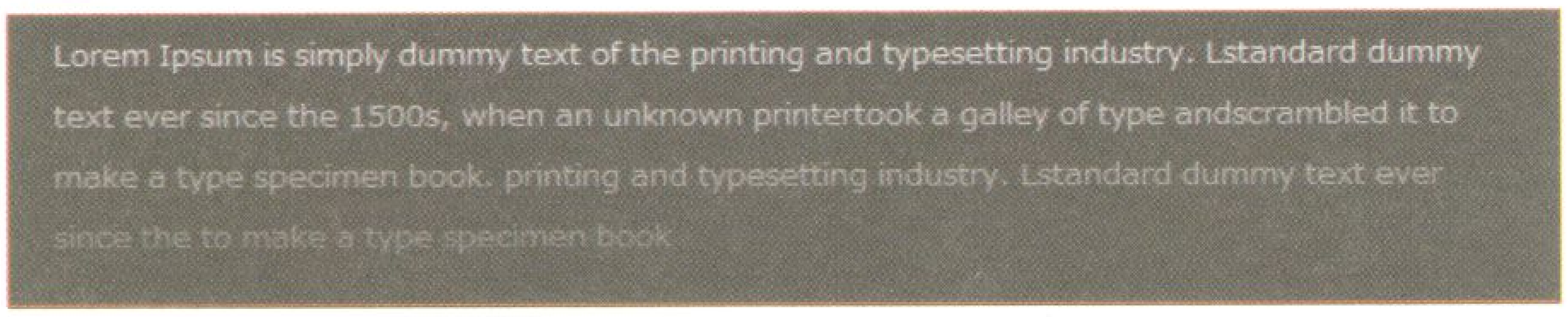
08

07で作成したアニメーションを、maskプロパティで読み込むようにします 08-1 。
これで、文字列がだんだん薄くなっていくような表示になります 08-2 。

08-1 画面幅に応じた設定ファイルを読み込む

```
p {
  -webkit-mask-image: -webkit-linear-gradient(top, rgba(0, 0, 0, 1), rgba(0, 0, 0, 0.2));
}
```

08-2 合成結果



まとめ [CONCLUSION]

非対応ブラウザのことを考慮すると、単に画像で切り出すほうが手軽かもしれません。
しかし、普通に切り出した画像だけでは難しい表現も、マスクを使えば手軽にできます。
仕組みも背景画像を重ねるように使うだけなので、簡単に実装できます。動きやインタラクティブ性のあるアクセントをつけてみましょう。

04 Webフォントでテキストを華やかに彩る



Webフォントはブラウザ上での文字の美しい表示や、画像でしかできなかった表現を実現可能にします。規格、ライセンス、クロスブラウザ対策など、実際の使用には問題点が多いのが現状ですが、Google Web Fonts APIを利用することで、導入の障壁が低くなります。ここではGoogle Web Fonts APIを使ったWebフォントの活用例を紹介します。

制作・文／中西祐介
（株式会社ワン・トゥー・テン・デザイン）

制作のポイント

- Webフォントの現状を把握する
- Google Web Fonts APIの利用
- CSS3を使って文字を装飾

対応ブラウザ

ブラウザ	対応
IE	P.E.
Safari	5以上
Firefox	P.E.
Chrome	10以上

※JavaScript使用時はIE6から対応（一部の効果には非対応）。

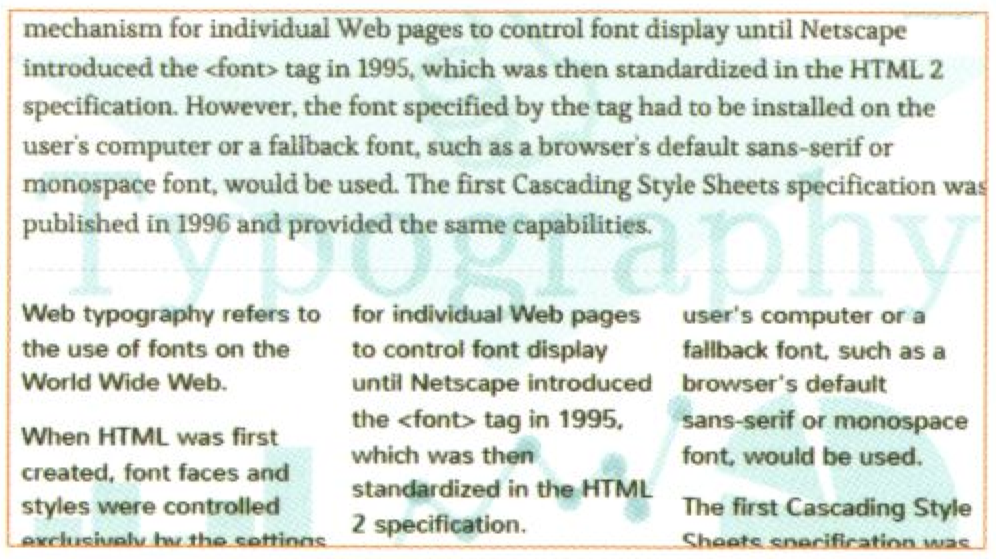
TECHNIC 1
テキストの表現の幅を広げる

日本語のWebフォントの充実は今後に期待するところですが、英数字のフォントは現在でも多くの選択肢があり、使用することで表現の幅が大きく広がります。



TECHNIC 2
画像を使ったような表現がテキストで可能に

今まで画像でしかできなかった表現も、テキスト（Webフォント）とCSS3で可能になります（例えば、作例の背景で使っている「Typography」の文字は画像ではなくテキストです）。





Webフォントの現状

01

フォントの指定は、今までも 01-1 のようにfont-familyプロパティを使うことで実現できました。しかし、サイトを見るすべてのユーザーが同じフォントをコンピューターにインストールしていない限り、同じ見た目にはなりません。

Webフォントのメリットはまさにそこにあり、フォントがインストールされていない環境でもサーバーに置いたフォントを使うことで、ユーザーの環境に左右されることなく同じ見た目が実現できます。

Webフォントで利用できるフォントファイルの主要なフォーマットには、以下のようなものがあります。

●TTF（TrueType）／OTF（OpenType）
デスクトップアプリケーションで使われているフォーマット。最も手軽に導入できるが、フォント制作者が生のデータをサーバーに上げることに難色を示したため、WOFFにその座を明け渡すことに。

●EOT（Embedded OpynType）
MicrosoftがWebフォント向けに提唱。Microsoftが提供している専用コンバートツール「Web Embedding Fonts Tool（WEFT）」で、EOTファイルを作成することができる。

●WOFF（Web Open Font Format）
Mozillaが中心となって提案を行い、ほかのベンダーからも賛同を得て、W3CにWebFonts Woking Groupが設立された。そしてWebFonts Working Groupで仕様策定が進められ、WOFFはW3C勧告候補になった。そのため、今後のWebフォントの標準フォーマットになることが確定している。
基本的にWebサイトでのみ利用でき、デスクトップアプリケーションでは利用できないとしてライセンスを分離している。

●SVG Fonts
SVG（Scalable Vector Graphics）形式の画像を@font-faceとfont-familyで定義して、文字として表示させたもの。SVGはXMLベースのベクターグラフィックス言語で記述された画像フォーマットで、Illustratorなどを使ってSVG形式で保存することができる。

また、古いブラウザの対応するフォントファイルのフォーマットは 01-2 のようにバラバラです。

01-1 font-familyによるフォント指定

```
body {
  font-family: "ヒラギノ角ゴ Pro W3", "Hiragino Kaku Gothic Pro", sans-serif;
}
```

01-2 Webフォントのブラウザ互換性

対応ブラウザ	TTF／OTF	EOT／EOT Lite	WOFF	SVG
IE 4～5	×	○	×	×
IE 6～7	×	○	×	×
IE 8	×	○	×	×
IE 9以上	×	○	○	×
Firefox 3.5	○	×	×	×
Firefox 3.6	○	×	○	×
Firefox 3.7以上	○	×	○	○
Safari 3.1～5	○	×	×	○
Safari 5.1以上	○	×	○	○
Chrome 4～5	○	×	×	○
Chrome 6以上	○	×	○	○
Opera 10	○	×	×	○
Opera 11以上	○	×	○	○
iOS 3.x～4.0	×	×	×	○
iOS 4.2以上	○	×	×	○
iOS 5以上	○	×	○	○

Webフォントを使うには@font-faceでフォントファイルを指定しますが、各ブラウザに対応するには01-3のように複数のフォーマットを指定する必要があります。
ここ最近では、今後Webフォントの標準フォーマットとなることが確定しているWOFFが注目されています。このフォント形式には3つの利点があります。

・フォントデータが圧縮されているため、読み込みが速い。
・XMLベースの情報が付加されている。これにはフォントのファウンダリ（開発会社）名やURL、著作権などの細かな情報などが盛り込まれているため、ライセンス情報が明白。
・新しいフォーマットで基本的にオープンソースであるため、同じくオープンソース開発であるFirefoxなどにとって扱いやすい。

対応ブラウザが最近のものだけでよければ、01-4のようにWOFF形式だけを指定します。
そのほかに、注意しなくてはならない点はやはりライセンスの問題です。サーバーにフォントファイルをアップロードしCSSでファイルのパスを指定して利用するため、フォントファイルを勝手にダウンロードすることができます。このようにリスクが大きいため、フォントファウンダリ（開発会社）側が指定するライセンスに細心の注意をはらう必要があります。
Webフォントは使いようによっては表現の幅が大きく広がるおもしろい技術です。しかし、アンチエイリアスはブラウザ依存で、日本語フォントは英数字に比べ文字数が多いためデータ量が大きくなることと、ライセンスの問題で選べる数が少なくなってしまいがちです。こういったデメリットを考慮した上で導入する必要があります。

01-3 複数フォーマットのフォント指定

```
@font-face {
  font-family: 'example';
  src: url(GraublauWeb.eot); /* IE用 */
  src: local('example'), /* フォントがローカルにインストールされている場合のための指定 */
    url('example.otf') format('opentype'); /* Firefox, Opera, Safari用 */
}
body {
  font-size: 14px;
  font-family: example; /* @font-faceで定義したフォント名を指定 */
  line-height: 1.5;
}
```

01-4 WOFF形式だけのフォント指定

```
@font-face {
  font-family: 'example';
  url('example.woff') format('woff'); /* woff形式 (IE9, Firefox3.6+, Chrome6+, Safari5.1+, Opera11.10+) */
}
body {
  font-size: 14px;
  font-family: example; /* @font-faceで定義したフォント名を指定 */
  line-height: 1.5;
}
```

Google Web Fontsの使い方

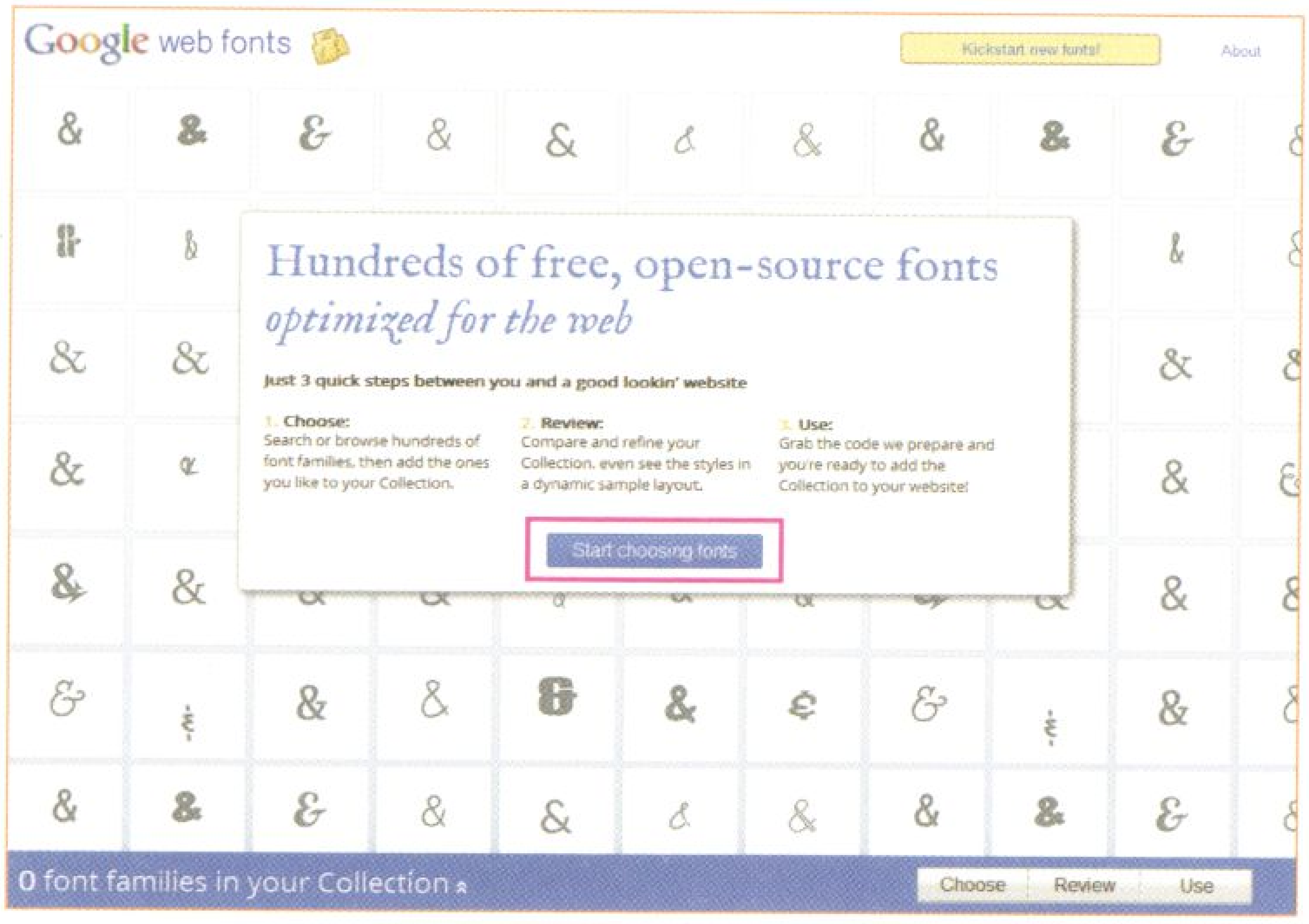
02

前述したクロスブラウザ、ライセンスなどの問題点を考慮すると、Webフォントは導入障壁が高いです。そこで、Google Web Fonts APIの出番となります。このAPIを利用すると、アクセス時に利用したブラウザに適したフォントファイルを適用してくれます。
クロスブラウザをクリアにし、ライセンスも明記されているため、それらに則れば問題ありません。日本語フォントがないというデメリットがありますが、導入障壁を下げてくれることは言うまでもありません。
Google Web Fonts APIはWebで扱うフォントをホスティングするサービスで、利用したいフォントを選ぶだけで、HTMLとCSSに記載するコードを生成してくれます。利用するには、まず02-1のURLにアクセスし、「Start choosing fonts」ボタンをクリックします02-2。

02-1 Google Web Fonts

http://www.google.com/webfonts/#HomePlace:home

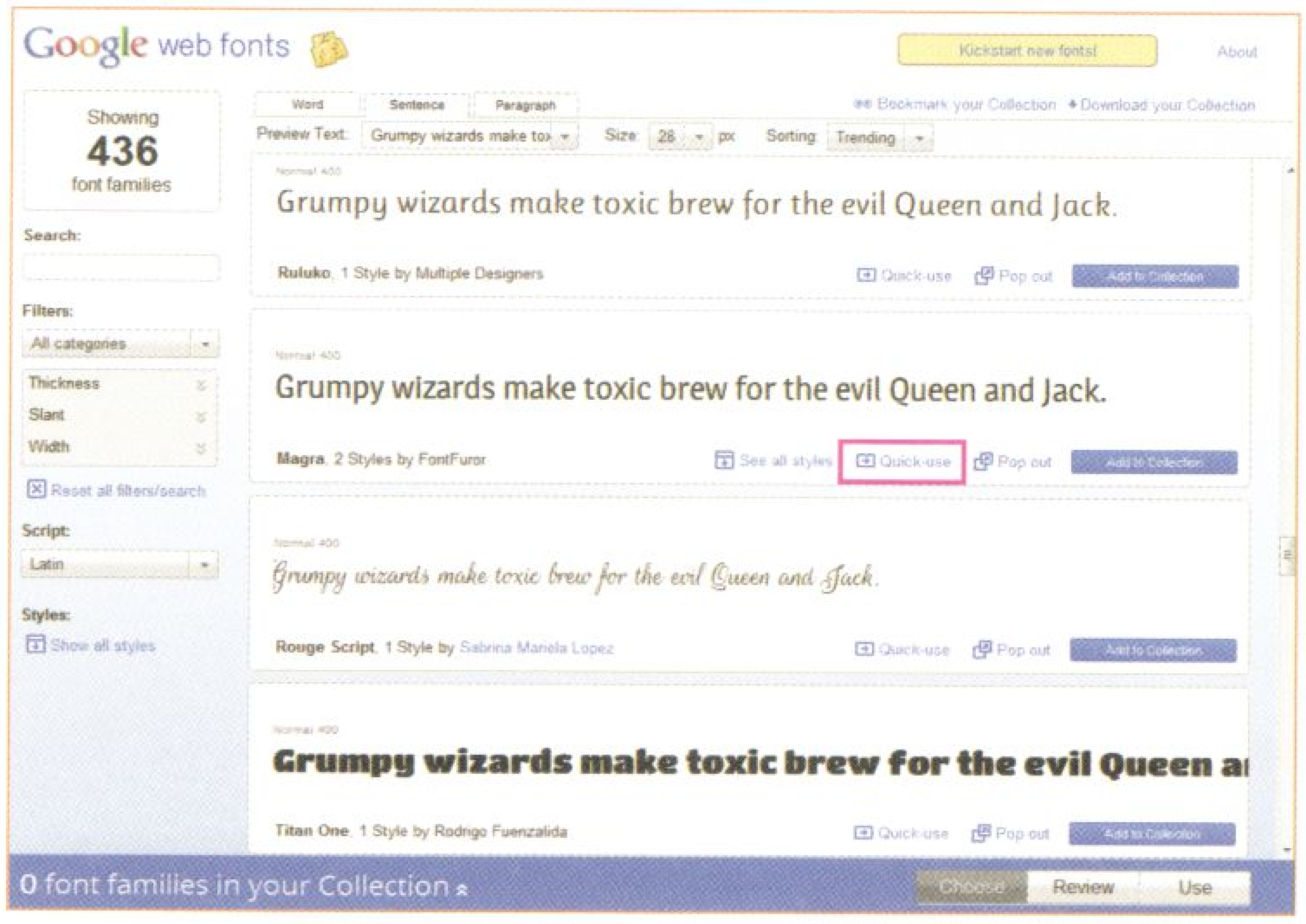
02-2 Google Web Fontsの利用開始画面



03

03-1 一覧からフォントを選択

次の画面で多くのフォントの一覧が表示されるので、利用したいフォントを選んで「Quick-use」ボタンをクリックします03-1。

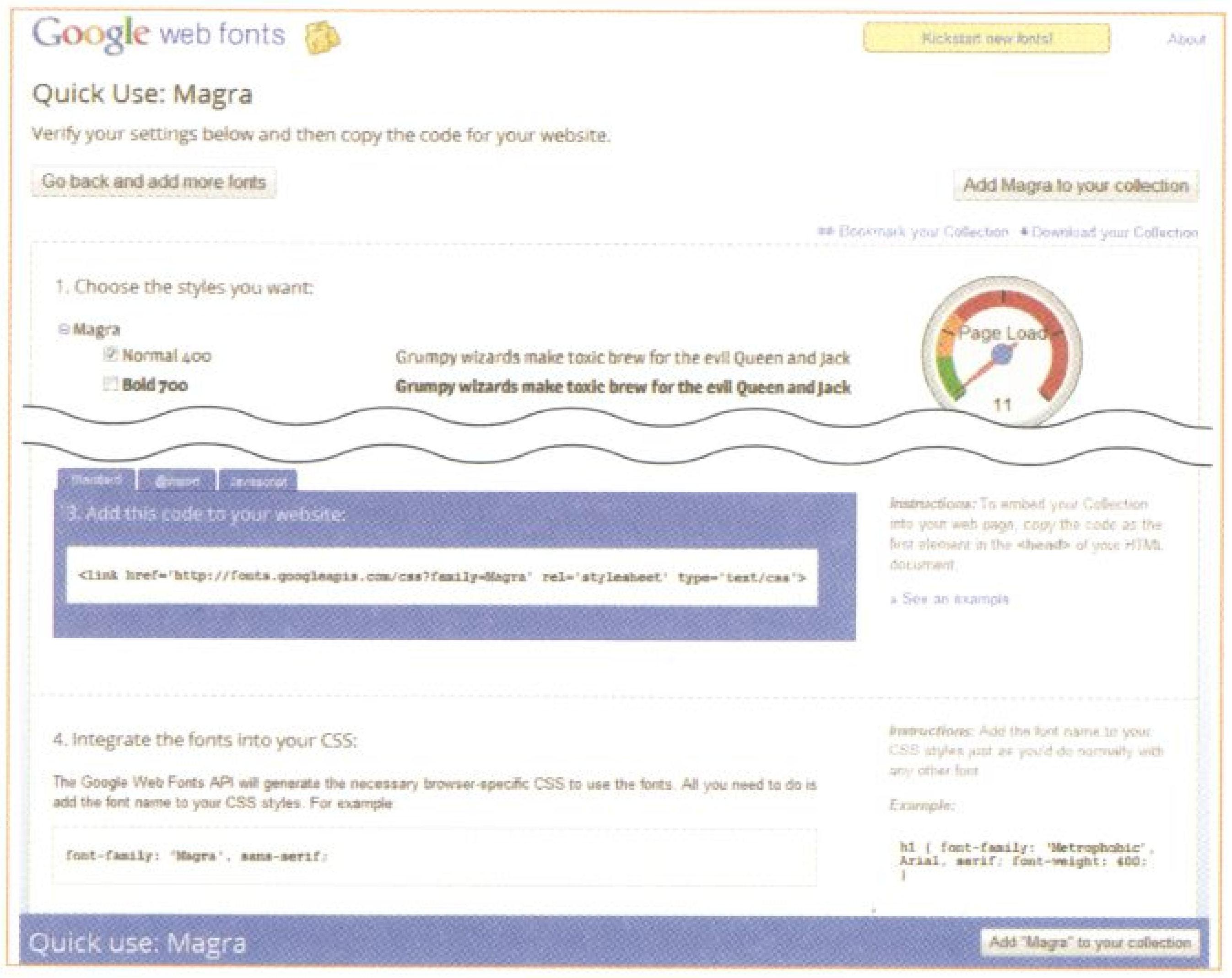


04

04-1 フォントの設定画面

04-1のようなフォントの設定画面が表示されるので、「1. Choose the styles you want:」で使うフォントの太さを選択します。

「3. Add this code to your website:」の欄にはHTMLのheadに記述するコード（CSSファイルのリンク）が、「4. Integrate the fonts into your CSS:」の欄にはCSSに記述するコード（font-family）が生成されます。



MEMO

フォントの種類によっては、「1. Choose the styles you want:」で太さを選択できません。また、複数のサブセットが含まれるフォントの場合は、「2. Choose the character sets you want:」でサブセットを選択することもできます。

05

04で生成されたCSSファイルのリンク 05-1 をコピーして、HTMLのheadに貼り付けます。

05-1 CSSファイルのリンク

```
<link href='http://fonts.googleapis.com/css?family=Magra' rel='stylesheet' type='text/css'>
```

TIPS

01のように「&text=abcde」と付けると、「Magra」というフォントの「abcde」データのみダウンロードし、適用することができます。Webフォントを適用する文字が決まっている場合は容量の削減になるので、こういった方法が有効です。

01 「Magra」フォントの「abcde」データのみダウンロード

```
<link href='http://fonts.googleapis.com/css?family=Magra&text=abcde'
rel='stylesheet' type='text/css'>
```

06

「Fugaz One」と「Magra」のように複数のフォントを使う場合には、06-1 のようにフォント名とフォント名を「|」(半角のvertical bar)で区切ります。

06-1 複数のフォントを使う場合

```
<link href='http://fonts.googleapis.com/
css?family=Fugaz+One|Magra'
rel='stylesheet' type='text/css'>
```

ATTENTION!

フォントを指定する際に注意すべき点は、「Fredericka the Great」のようにフォント名にスペースのあるフォントを指定する場合、01のように余白を「+」に変更するという点です。モダンブラウザでは余白のままでも問題ありませんが、IEの古いバージョンでは認識してくれず、フォントを適用してくれません。

01 余白のあるフォントの指定

```
// 間違ったマークアップ
<link href='http://fonts.googleapis.com/css?family=Fredericka the Great'
rel='stylesheet' type='text/css'>

// 正しいマークアップ
<link href='http://fonts.googleapis.com/css?family=Fredericka+the+Great'
rel='stylesheet' type='text/css'>
```

07

次に、04で生成されたfont-familyをコピーしてCSSに貼り付け 07-1、HTML中のGoogle Web Fontsを使用したい箇所にスタイルを適用します 07-2。

Google Web Fontsの使い方は以上です。

07-1 CSSにfont-familyを指定

```
p.text {
  font-family: 'Magra', sans-serif;
  font-size: 30px;
}
```

07-2 HTMLの使用したい箇所に適用

```
<p class="text">ここがMagraというフォントになります。</p>
```

ATTENTION!

IEでは、ローカル上で確認するとフォントが正常に表示されない場合があります。



Google Web Fontsを利用したページをつくる

08

ここからGoogle Web Fontsを使用したサンプルの解説に入ります。
サンプルはHTML5を使って作成していますが、IE8以下の古いブラウザはHTML5に対応していません。
そこで、ここでは「html5.js」08-1 というJavaScriptファイルを読み込むことでIE6～8にもHTML5の要素が適用できるようにします。08-2 のように入力し、ここでは「scripts」というフォルダにhtml5.jsを格納しました。
<!--[if lte IE 8]>～<![endif]-->で囲っているのは条件分岐コメントで、HTML5に対応していないIE8以下でhtml5.jsのファイルを読み込むように指示しています。

08-1 html5.js

http://html5shim.googlecode.com/svn/trunk/html5.js

08-2 html5.jsの読み込み（index.html）

```
<!--[if lte IE 8]>
<script src="./scripts/html5.js" type="text/javascript"></script>
<![endif]-->
```

MEMO

「html5.js」については、INTRODUCTIONの「CSS3使用時の設計指針と注意点」（16ページ）もご覧ください。

09

次に、ブラウザによってCSSの各要素のデフォルト値が違うのを揃えるために、「html5reset-1.6.1.css」というファイルをダウンロードして読み込みます 09-1。HTMLファイルでは 09-2 のように記述しました。
このサンプルではGoogle Web Fontsを利用しているので、そのCSSへのリンクも記載します。

09-1 html5reset-1.6.1.css

http://code.google.com/p/html5resetcss/downloads/detail?name=html5reset-1.6.1.css&can=2&q=

09-2 CSSの読み込み（index.html）

```
<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=
Fugaz+One|Frijole|Nosifer|Cabin+Sketch|Miltonian|Lobster+Two">
<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=
Fredericka+the+Great|Limelight|Wellfleet|Esteban|Nunito|Noticia+Text">
<link rel="stylesheet" href="./styles/html5reset-1.6.1.css" type="text/css" />
<link rel="stylesheet" href="./styles/default.css" type="text/css" />
<link rel="stylesheet" href="./styles/content.css" type="text/css" />
<!--[if IE 9]>
<link rel="stylesheet" href="./styles/ie9.css" type="text/css" />
<![endif]-->
<!--[if lte IE 8]>
<link rel="stylesheet" href="./styles/ie6_7_8.css" type="text/css" />
<![endif]-->
```

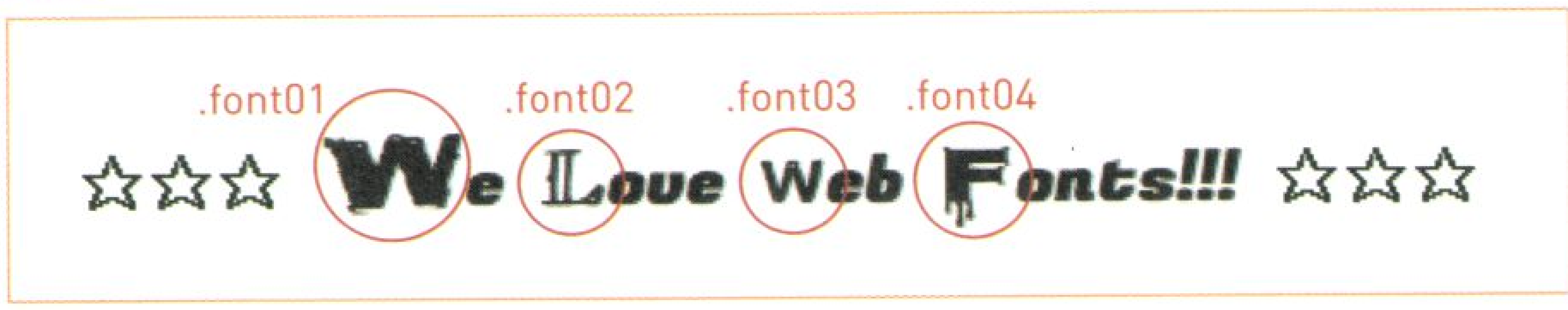
MEMO

「content.css」はメイン表示領域についての指定をまとめたCSSファイル、「default.css」はページ全体のレイアウトの大枠について記述したCSSファイルです。
また、IEでの表示を確保するためie9.css（IE9用）、ie6_7_8.css（IE6～8用）を作成して、CSSを上書き指定しています。これらのCSSを「styles」フォルダに格納して読み込ませています。

10

次に、ヘッダーとフッターにある「☆☆☆ We Love Web Fonts!!! ☆☆☆」の部分を見ていきましょう 10-1。どちらも同じ指定をしているので、ここではヘッダーを例に説明します。
p要素全体に「Fugaz One」フォントを指定し、各単語の1文字目だけを変えたいので、spanでくくりclassを付けます。そして、それぞれ別のfont-familyを指定しています 10-2 10-3。

10-1 単語の1文字目にそれぞれ別のフォントが適用されている



10-2 ヘッダー部分（index.html）

```
<p>☆☆☆ <span class="font01">W</span>e <span class="font02">L</span>ove <span class="font03">W</span>eb <span class="font04">F</span>onts!!! ☆☆☆</p>
```

10-3 ヘッダーに使うフォントの指定（content.css）

```
#demoArea header p {
  padding-bottom: 10px;
  text-align: center;
  font-family: 'Fugaz One', Serif;
  font-size: 20px;
  font-weight: bold;
  line-height: 0.8;
  border-bottom: 1px solid #dcdcdc;
  zoom: 1;
}
#demoArea header p span.font01 {
 font-family: 'Frijole', Serif;
 font-size: 30px;
}
#demoArea header p span.font02 {
  font-family: 'Miltonian', Serif;
  font-size: 25px;
}
#demoArea header p span.font03 {
  font-family: 'Cabin Sketch', Serif;
  font-size: 20px;
}
#demoArea header p span.font04 {
  font-family: 'Nosifer', Serif;
  font-size: 25px;
}
```

11

次に、ヘッダーの下にある「CSS3 Web Fonts」というh1要素の部分を見てみましょう。星やハートは背景画像として配置しています。ハートの画像の上にある「Web」というフォントを見ると、アルファベットの穴の開いている部分も背景画像がきちんと表示されていて、Webフォントで行える表現の素晴らしさを知ることができます 11-1。
ここではh1要素に単語ごとに異なるフォントを指定しています 11-2。
また、h1にposition: relative;として配置の基準を指定した上で、「CSS3」、「Web」、「Fonts」をposition: absolute;で絶対配置しました 11-3。

11-1 フォントの穴の開いている部分も背景画像が表示されている



11-2 「CSS3 Web Fonts」の部分（index.html）

```
<h1> <span class="font01">CSS3</span> <span class="font02">Web</span> <span class="font03">Fonts</span> <span class="font04"></span></h1>
```

ATTENTION!

Google Web Fontsを利用した場合でも、ブラウザによって文字の字送りや配置に微妙なズレがあるようです。本サンプルでもFirefox 9などの一部のブラウザでは、「CSS3」という文字と背景の星の位置がずれて表示されます。本サンプルではWebフォントの利便性をお伝えするためにあえて複雑な指定を試みていますが、実際に使用するときはブラウザによってある程度、表示のずれが生じることを考慮しましょう。



11-3 「CSS3 Web Fonts」の部分に使うフォントの指定（content.css）

```
#demoArea header h1 {
    position: relative;
    height: 165px;
    line-height: 0.8;
    background: url(../images/demoarea_
    back_02.gif) no-repeat 370px 15px;
}
#demoArea header h1 span {
    display: block;
    position: absolute;
}
#demoArea header h1 span.font01 {
    overflow: hidden;
    width: 340px;
    font-family: 'Lobster Two', Serif;
    font-size: 160px;
    top: 20px;
    left: 10px;
}
#demoArea header h1 span.font02 {
    overflow: hidden;
    width: 190px;
    font-family: 'Fredericka the Great', Serif;
    font-size: 90px;
    top: 25px;
    left: 350px;
}
#demoArea header h1 span.font03 {
    overflow: hidden;
    width: 190px;
    font-family: 'Limelight', Serif;
    font-size: 60px;
    top: 100px;
    left: 350px;
}
```

12

ヘッダーの左にある「2012」の部分は、border-radiusプロパティでボックスを円にしています 12-1 12-2。

ここでは、さらにtopとleftにネガティブな-35pxという値を指定することで、囲み枠から円が飛び出した表現にしました。

絶対配置にしているので、基準となる位置を指定するために#demoAreaにposition: relative;を指定しています 12-3。

12-1 「2012」の部分が円になっている



12-2 「2012」の部分（index.html）

```
<p id="zone01">2012</p>
```

12-3 「2012」部分のtopやleft、絶対配置の指定（content.css）

```
#demoArea {
    position: relative;
    — 省略 —
}
p#zone01 {
    position: absolute;
    top: -35px;
    left: -35px;
    width: 100px;
    height: 59px;
    padding-top: 41px;
    font-family: 'Wellfleet', Serif;
    font-size: 20px;
    line-height: 0.8;
    color: #fff;
    text-align: center;
    background: #000;
    -webkit-border-radius: 60px;
    -moz-border-radius: 60px;
    border-radius: 60px;
}
```

13

囲み枠の右上の「Font」、左下の「Style」、右下の「type」部分は、p要素の中のspanを、transformプロパティを用いて回転させています。ここでは、右上の「Font」を例に説明します。

p要素の高さと横幅を指定して、overflow: hidden;でそれ以外の範囲を表示させないようにしています 13-1 13-2。そうすることで、13-3のような表現が可能となります。

IEではtransformプロパティに対応していないため、非表示にしています。

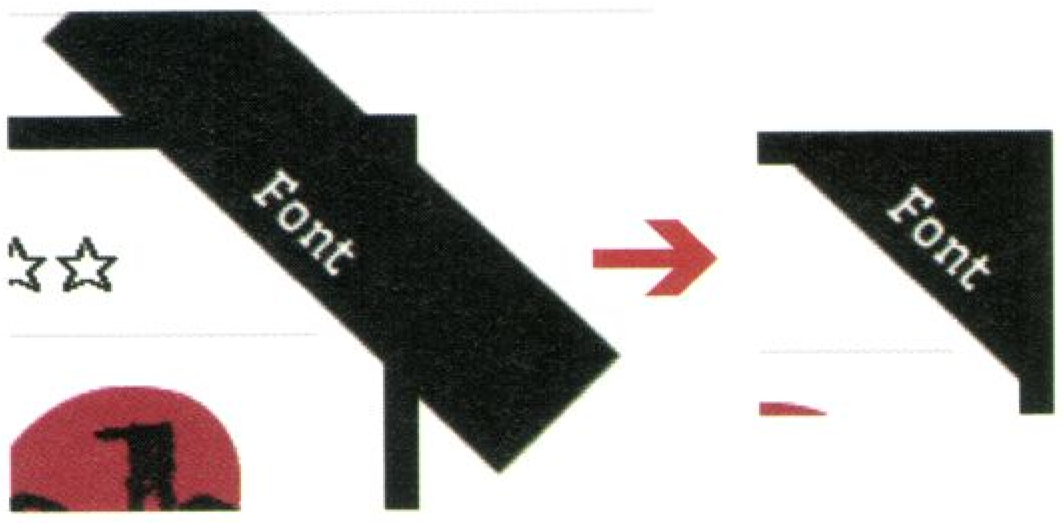
13-1 「Font」の部分（index.html）

```
<p id="zone02"><span>Font</span></p>
```

13-2 「Font」の部分（content.css）

```
p#zone02 {
  position: absolute;
  top: 0;
  right: 0;
  overflow: hidden;
  width: 75px;
  height: 75px;
}
p#zone02 span {
  display: block;
  width: 150px;
  margin: -10px 0 0 -35px;
  padding: 30px 20px 8px;
  font-family: 'Wellfleet', Serif;
  font-size: 16px;
  line-height: 0.8;
  color: #fff;
  text-align: center;
  background: #000;
  -moz-transform: rotate(45deg);
  -webkit-transform: rotate(45deg);
  transform: rotate(45deg);
}
```

13-3 高さと横幅の指定範囲外は表示されない

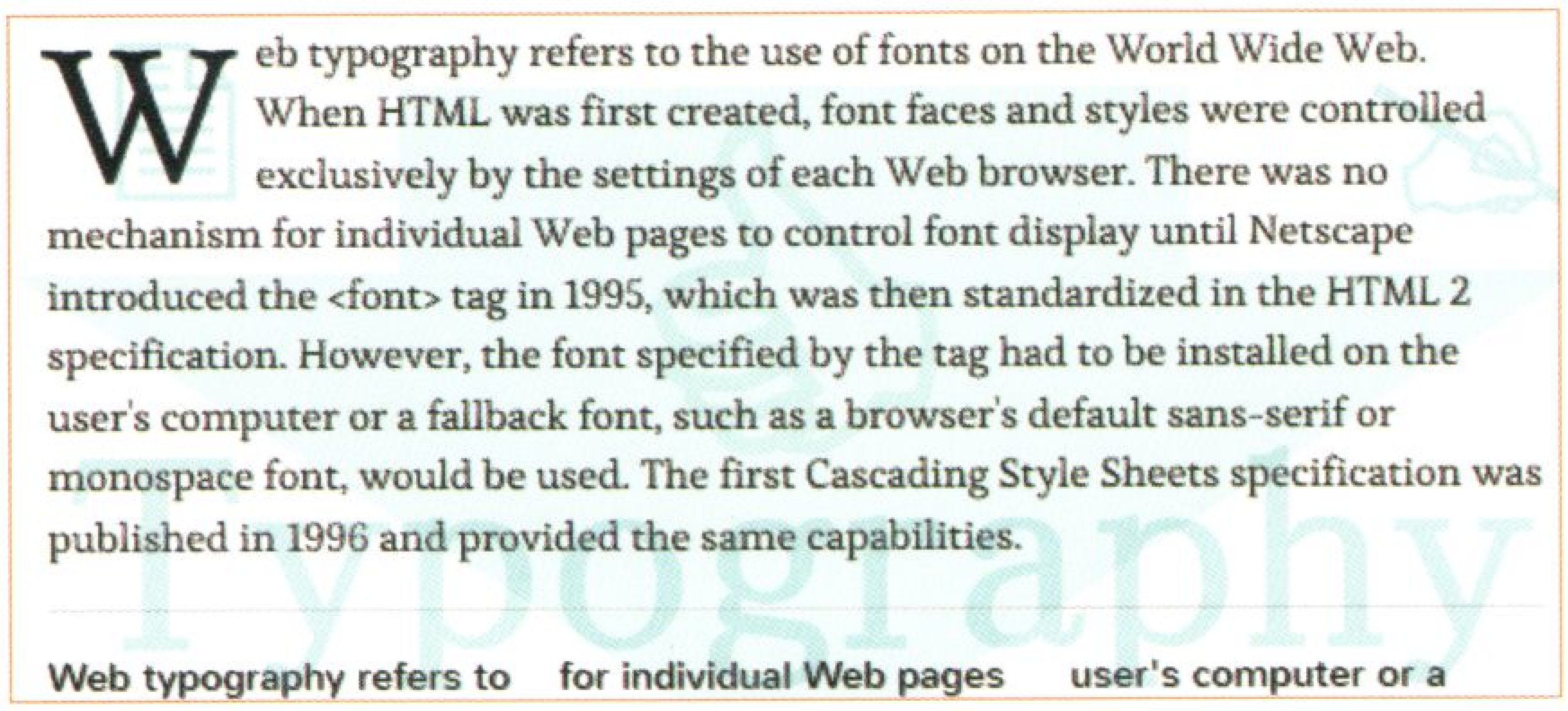


overflow: hidden;を指定しない場合　　overflow: hidden;を指定した場合

14

14-1 の DropCaps の表現は、:first-letter 擬似要素を用いて要素の最初の文字にスタイルを適用しました **14-2**。これは IE6 も対応しています。
最初の文字を大きくして配置するだけで、印象の違う見せ方が可能になります。

14-1 最初の1文字が大きく表示されている



Web typography refers to the use of fonts on the World Wide Web. When HTML was first created, font faces and styles were controlled exclusively by the settings of each Web browser. There was no mechanism for individual Web pages to control font display until Netscape introduced the <font> tag in 1995, which was then standardized in the HTML 2 specification. However, the font specified by the tag had to be installed on the user's computer or a fallback font, such as a browser's default sans-serif or monospace font, would be used. The first Cascading Style Sheets specification was published in 1996 and provided the same capabilities.

Typography

Web typography refers to　for individual Web pages　user's computer or a

14-2 要素の最初の文字にスタイルを適用（content.css）

```
p.text01 {
  margin-bottom: 15px;
  padding-bottom: 15px;
  font-family: 'Esteban', serif;
  font-size: 15px;
  line-height: 1.5;
  border-bottom: 1px solid #dcdcdc;
}
p.text01:first-letter {
  display: block;
  float: left;
  margin: 10px 7px 0 0;
  font-size: 70px;
  line-height: 0.8;
}
```

MEMO

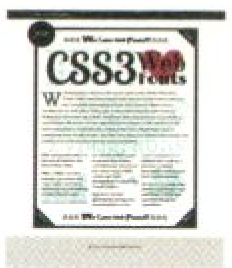
14-1 の背景にある「Typography」という文字は画像ではありません。Web フォントを使い opacity プロパティを利用することで透過させています **01** **02**。このように今まで画像でしかできなかった表現も、Web フォントと CSS3 を使うことでテキストでも可能になります。

01 「Typography」をテキストとして指定（index.html）

```
<div class="columns01">
  <h2>Typography</h2>
  ― 省略 ―
</div>
```

02 opacity プロパティの指定（content.css）

```
div.columns01 h2 {
  position: absolute;
  top: -100px;
  left: 0px;
  width: 100%;
  overflow: hidden;
  font-family: 'Noticia Text', serif;
  font-size: 95px;
  font-weight: bold;
  line-height: 1.5;
  color: #09c;
  z-index: 0;
  zoom: 1;
  -ms-filter:"progid:DXImageTransform.
  Microsoft.Alpha(Opacity=20)";
  filter: alpha(opacity=20);
  -moz-opacity:0.2;
  -khtml-opacity: 0.2;
  opacity: 0.2;
}
```



15

次に、DropCapsの下にある英文が3段組みになった部分を見てみましょう。columns-countプロパティを指定することで、文章の段組みが可能になります。このサンプルでは、columns-count: 3;と指定して横3列に文字を配置しました 15-1 15-2。文字量に合わせて高さを自動的に算出してくれるため、崩れることもありません。

IEはcolumns-countプロパティに対応していないため 15-3 15-4、段組みの表示は崩れてしまいますが、文章の可読性は保持されるので、IEでの表示に注意しながら有効活用してください。

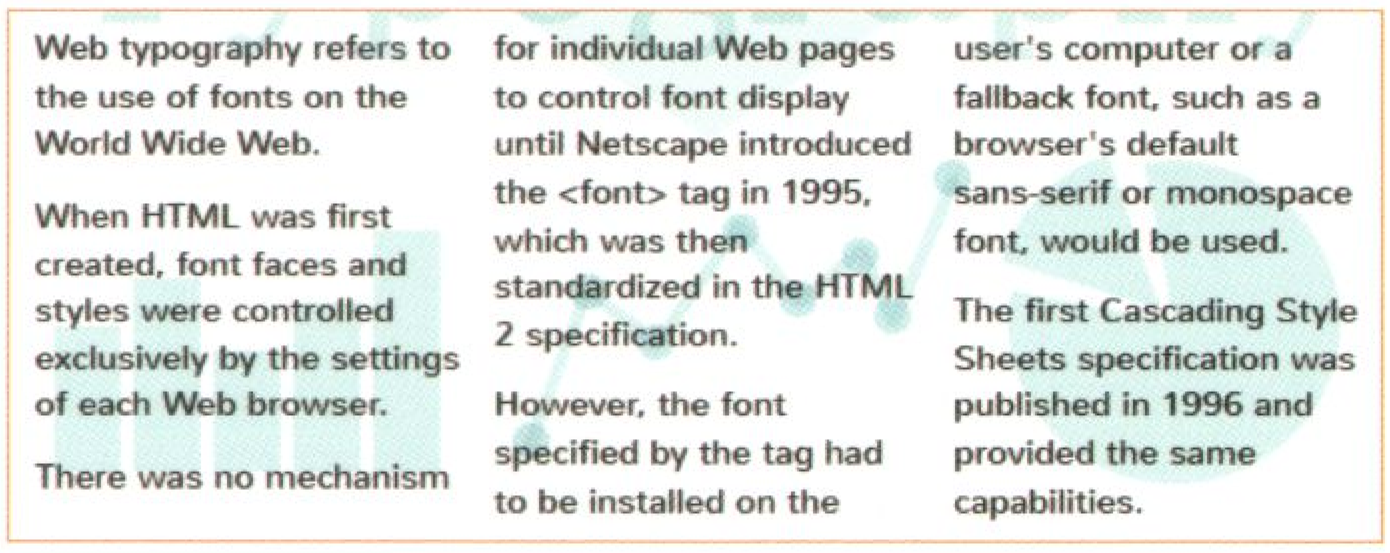
15-1 段組みにする文章（index.html）

```
<div class="columns01">
<h2>Typography</h2>
<p>Web typography refers to the use of fonts on the World Wide Web.</p>
<p>When HTML was first created, font faces and styles were controlled exclusively by the settings of each Web browser.</p>
<p>There was no mechanism for individual Web pages to control font display until Netscape introduced the &lt;font&gt; tag in 1995, which was then standardized in the HTML 2 specification.</p>
<p>However, the font specified by the tag had to be installed on the user's computer or a fallback font, such as a browser's default sans-serif or monospace font, would be used.</p>
<p>The first Cascading Style Sheets specification was published in 1996 and provided the same capabilities.</p>
</div>
```

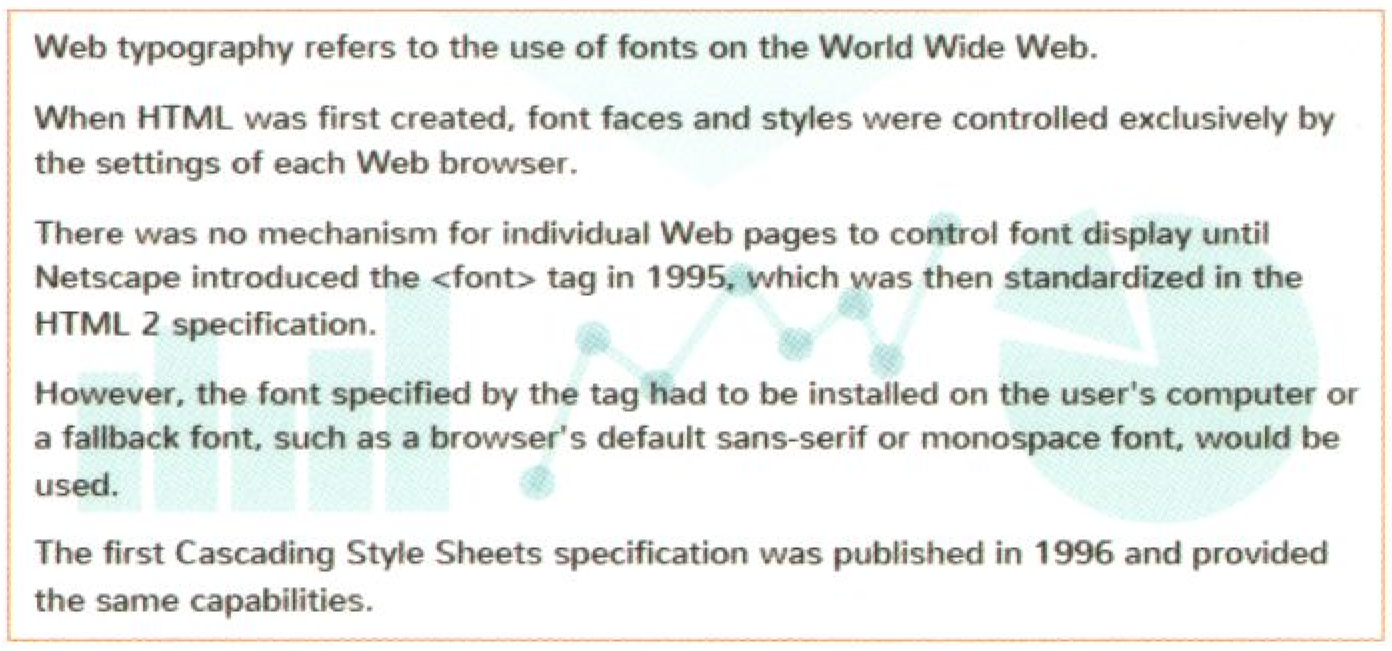
15-2 段組みの指定（content.css）

```
div.columns01 {
  position: relative;
  width: 100%;
  -moz-column-count: 3;
  -webkit-column-count: 3;
  columns-count: 3;
  border-bottom: 1px solid #fff;
}
```

15-3 Firefox、Chrome、Safariでは段組みで表示される



15-4 IEでは段組みで表示されない

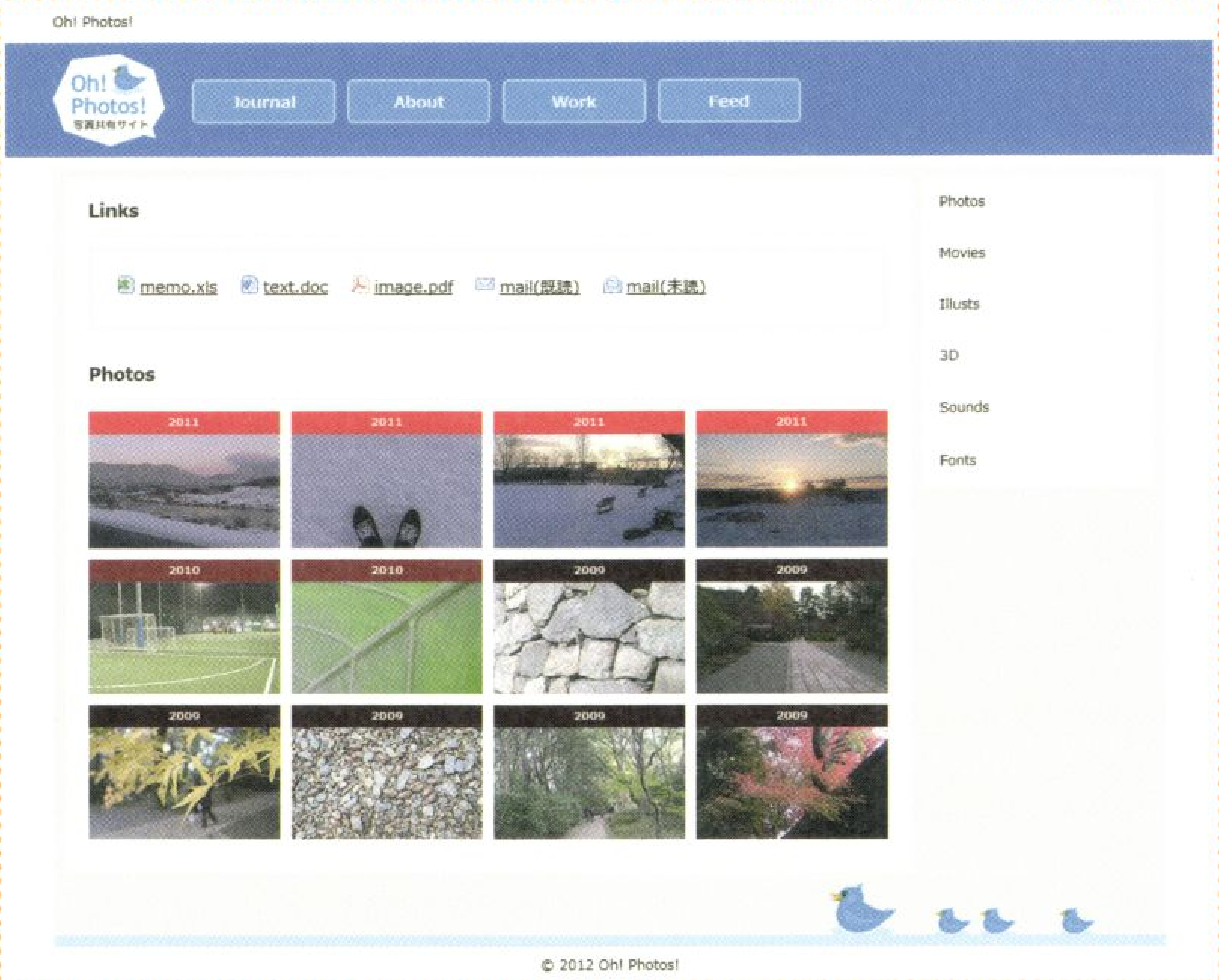


まとめ [CONCLUSION]

Webフォントは普通のテキストと同じように扱えるため、text-shadowプロパティやtransformプロパティなどで影付けや回転させることもできますし、ほかのCSS3プロパティを組み合わせることでいろいろな表現が可能です。

ここではWebフォントの活用例を紹介する意図から多くのフォントを使いましたが、Webフォントを多用したページは読み込むまでに少し時間がかかります。実際に使う際にはタイトルだけに使うなど、より効果が高いポイントに絞り込んで使う必要があるでしょう。

05 属性セレクタで写真を装飾したギャラリー



CSS3で追加された属性セレクタを駆使して、リンクにアイコン画像を表示したり、写真に視覚的な装飾を施したりできます。ここではIE6～8でも見た目を整えるためにJavaScriptを利用しました。

制作・文／中西祐介
（株式会社ワン・トゥー・テン・デザイン）

制作のポイント

- 属性セレクタの条件「^」、「$」、「*」の使い分け
- 属性セレクタを考えたマークアップ
- IEでも見た目を揃えるためにJavaScriptで補う

対応ブラウザ

IE	P.E.
Safari	5 以上
Firefox	4 以上
Chrome	10 以上

※JavaScript使用時はIE6から対応。
（IE6～8は:after、contentに非対応）

TECHNIC 1

ファイルの種類別にアイコンを表示

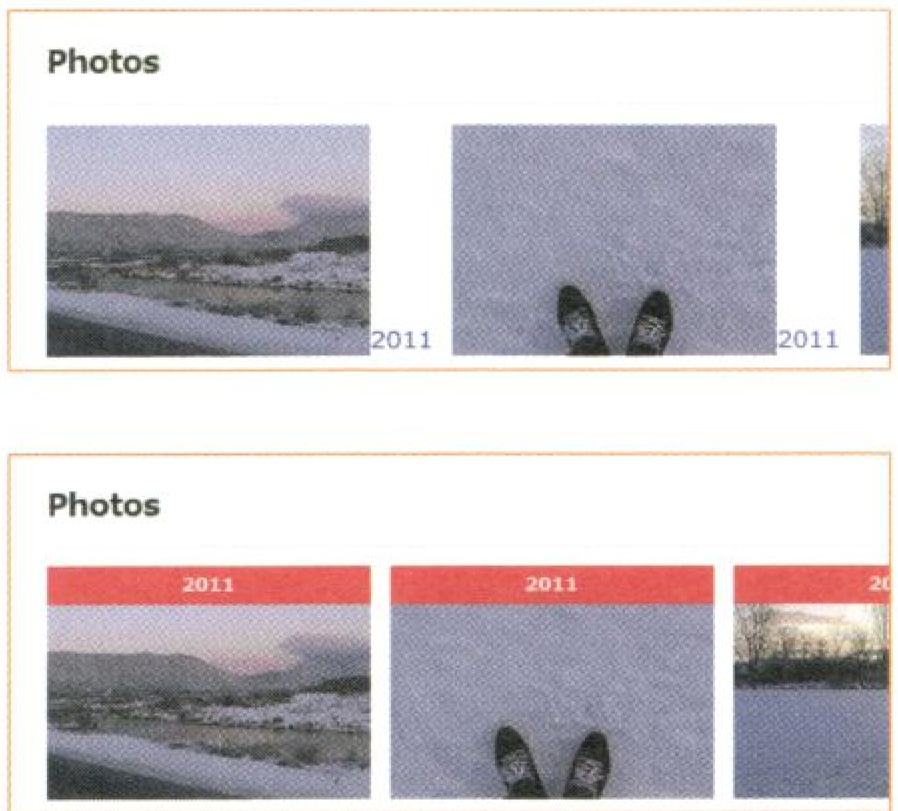
属性セレクタの3つの条件を使い分けて、特定の属性の値が指定した条件で始まっている／終わっている場合などにCSSを適用しています（画面左はCSS適用前、右は適用後）。



TECHNIC 2

要素の直後に内容を挿入する

:after擬似要素とcontentプロパティによって、写真の直後にテキストを挿入しています（画面下はテキストを絶対配置で、写真の上に配置したもの）。





ベースのHTMLを作成する

01

まずは、ベースになるHTMLについて解説します。サンプルはHTML5で作成しています。
IE8以下の古いバージョンはHTML5に対応していないため、「html5.js」01-1 というJavaScriptファイルを読み込むことでIE6〜8にもHTML5の要素が適用できるようにしました。「scripts」というフォルダにhtml5.jsを格納し、index.htmlに 01-2 のように記述します。<!--[if lte IE 8]> 〜 <![endif]-->で囲っているのは条件分岐コメントで、HTML5に対応していないIE8以下でhtml5.jsのファイルを読み込むように指示しています。

01-1 html5.js

```
http://html5shim.googlecode.com/svn/trunk/html5.js
```

01-2 html5.jsの読み込み（index.html）

```
<!--[if lte IE 8]>
<script src="./scripts/html5.js" type="text/javascript"></script>
<![endif]-->
```

02

次に、ブラウザによってCSSの各要素のデフォルト値が違うのを揃えるために、「html5reset-1.6.1.css」というファイルをダウンロードして読み込みます 02-1。HTMLファイルでは 02-2 のように記述しました。

MEMO

ページ全体のレイアウトの大枠については「default.css」に、リンクや写真を表示する領域についての指定は「content.css」に記述しています。これらを「styles」フォルダに格納して読み込ませています。「ie6_7_8.js」と「ie6.js」については後述します。

02-1 html5reset-1.6.1.css

```
http://code.google.com/p/html5resetcss/downloads/
detail?name=html5reset-1.6.1.css&can=2&q=
```

02-2 CSSの読み込み（index.html）

```
<!--[if lte IE 8]>
<script src="./scripts/html5.js" type="text/javascript"></script>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.
min.js"></script>
<script src="./scripts/ie6_7_8.js" type="text/javascript"></script>
<![endif]-->
<!--[if IE 6]>
<script src="./scripts/ie6.js" type="text/javascript"></script>
<![endif]-->
<link rel="stylesheet" href="./styles/html5reset-1.6.1.css" type="text/css" />
<link rel="stylesheet" href="./styles/default.css" type="text/css" />
<link rel="stylesheet" href="./styles/content.css" type="text/css" />
```

href属性を使ってリンクに画像を表示

03

まずは、リンクしているファイルの種類に応じて画像を表示している「Links」の部分から見ていきます 03-1。HTMLを見ていただいてもわかるように、マークアップは普通のulのリストで、li要素にa要素でリンクが指定されています 03-2。

03-1 ブラウザで表示させたところ（CSS適用前）



03-2 「Links」の部分（index.html）

```
<section id="demoArea01">
  <ul>
    <li><a href="./memo.xls" target="_blank">memo.xls</a></li>
    <li><a href="./text.doc" target="_blank">text.doc</a></li>
    <li><a href="./image.pdf" target="_blank">image.pdf</a></li>
    <li><a href="./mail/001/" title="alreadyRead">mail(既読)</a></li>
    <li><a href="./mail/002" title="unread">mail(未読)</a></li>
  </ul>
</section>
```

04

ここではファイルに応じたアイコンを表示させるようにします。04-1では、href属性の内容が「.xls」、「.doc」、「.pdf」のどの拡張子で終わっているかを判定し、それぞれにCSSを適用してアイコン画像を表示しています。また、title属性の内容が「already Read」もしくは「unread」を含むかどうかを判定しています。

04-1 属性セレクタの条件でスタイルを適用（content.css）

```
#demoArea01 ul li a[href$=".xls"] {
  background: url(../images/demoarea_icon_01.gif) no-repeat left 0;
}
#demoArea01 ul li a[href$=".doc"] {
  background: url(../images/demoarea_icon_02.gif) no-repeat 0 0;
}
#demoArea01 ul li a[href$=".pdf"] {
  background: url(../images/demoarea_icon_03.gif) no-repeat 0 0;
}
#demoArea01 ul li a[title*="alreadyRead"] {
  background: url(../images/demoarea_icon_04.gif) no-repeat 0 0;
}
#demoArea01 ul li a[title*="unread"] {
  background: url(../images/demoarea_icon_05.gif) no-repeat 0 0;
}
```

04-2 ブラウザで表示させたところ（CSS適用後）



MEMO

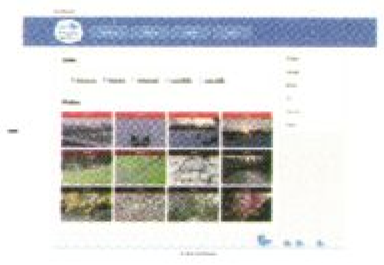
属性セレクタには条件が3つあります。「^」は指定した条件で始まっている場合、「$」は指定した条件で終わっている場合、「*」は指定した条件が含まれている場合です。01と02に構文と具体的な例を示します。

01 属性セレクタの条件

条件	構文	意味
^	E[foo^="bar"]	foo属性の値がbarで始まっている要素
$	E[foo$="bar"]	foo属性の値がbarで終わっている要素
*	E[foo*="bar"]	foo属性の値にbarを含む要素

02 属性セレクタの条件の具体例

例	意味
div[class^="abc"]	div要素のclass属性の値がabcで始まっている要素
p[title$="abc"]	p要素のtitle属性の値がabcで終わっている要素
img[alt*="abc"]	img要素のalt属性の値にabcを含む要素



05

IE6は属性セレクタに対応していないため、JavaScriptの正規表現を使って対策をとります05-1。href属性の内容が指定した条件（「.xls」、「.doc」、「.pdf」で終わっているか）を判定したり、title属性に「alreadyRead」もしくは「unread」を含むかを判定し、条件を満たせばCSSを指定しています。（「ie6.js」で記述したコードは筆者が作成したものなので、自由に改変していただいて結構です。）

05-1 IE6対策のjsファイル（ie6.js）

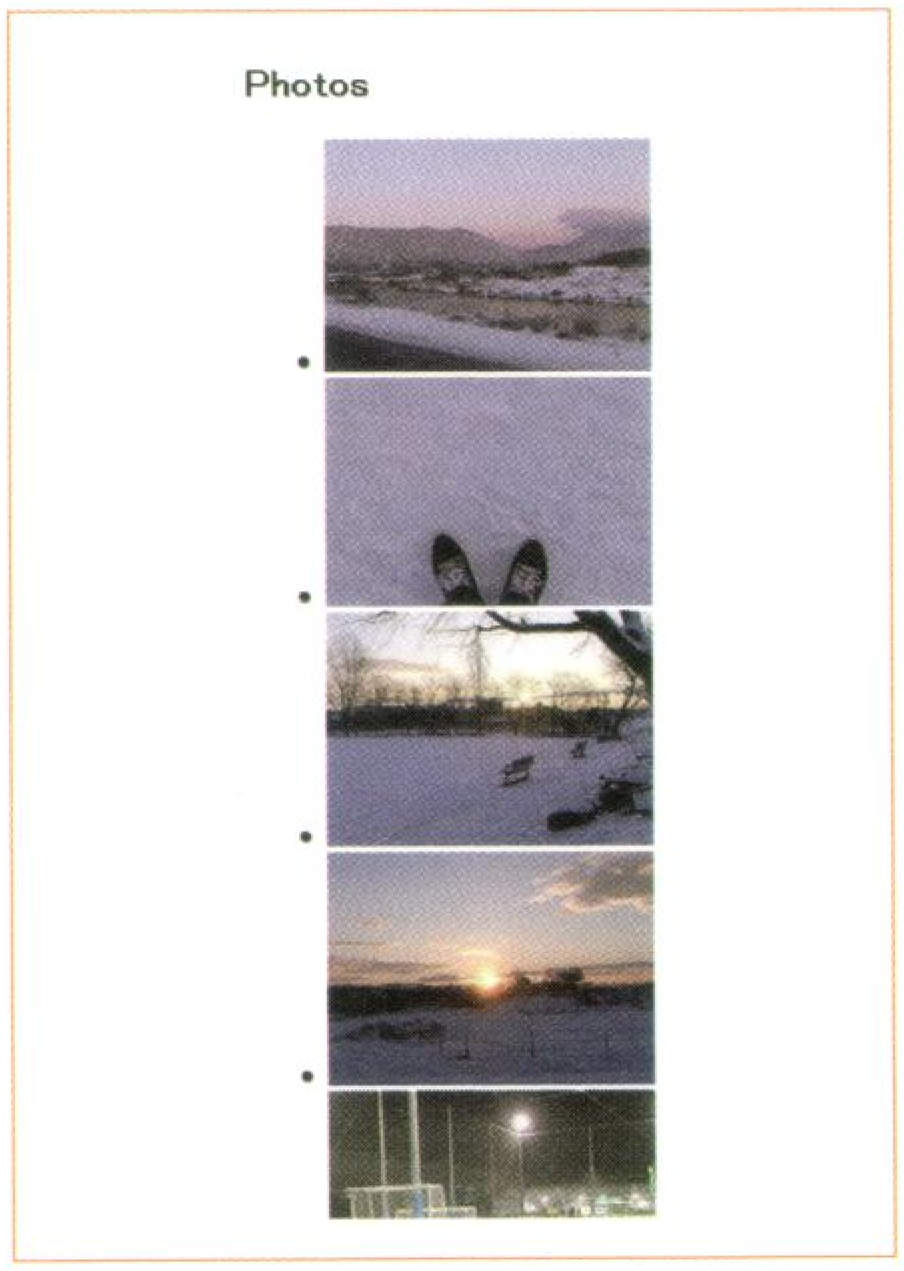
```
jQuery(function(){
  $('#demoArea01 li').each(function() {
    var linkHref = $(this).children('a').attr('href');
    var linkTitle = $(this).children('a').attr('title');
    if(linkHref) {
      if(linkHref.match(/.xls$/)) { $(this).children('a').css({ 'background':
      'url(./images/demoarea_icon_01.gif) no-repeat left 0' }); }
      if(linkHref.match(/.doc$/)) { $(this).children('a').css({ 'background'
      : 'url(./images/demoarea_icon_02.gif) no-repeat left 0' }); }
      if(linkHref.match(/.pdf$/)) { $(this).children('a').css({ 'background'
      : 'url(./images/demoarea_icon_03.gif) no-repeat left 0' }); }
    }
    if(linkTitle) {
      if(linkTitle.match(/alreadyRead/)) { $(this).children('a').css({
      'background': 'url(./images/demoarea_icon_04.gif) no-repeat left
      0' }); }
      if(linkTitle.match(/unread/)) { $(this).children('a').css({
      'background': 'url(./images/demoarea_icon_05.gif) no-repeat left
      0' }); }
    }
  });
});
```

写真の上にテキストを配置する

06

次に、写真の配置や装飾に入ります。03と同じく、ここもマークアップは至って普通で、ul要素のリストに写真とその写真へのリンクを張ったものです06-1。ブラウザで表示させると06-2のようになります。

06-2 ブラウザで表示させたところ



06-1 「Photos」の部分（index.html）

```
<section id="demoArea02">
  <ul>
    <li><a href="./images/2011/01/photo04_001.jpg"><img src="./images/2011/01/
    photo04_001.jpg" alt="" /></a></li>
    <li><a href="./images/2011/01/photo04_002.jpg"><img src="./images/2011/01/
    photo04_002.jpg" alt="" /></a></li>
    <li><a href="./images/2011/01/photo04_003.jpg"><img src="./images/2011/01/
    photo04_003.jpg" alt="" /></a></li>
    <li><a href="./images/2011/01/photo04_004.jpg"><img src="./images/2011/01/
    photo04_004.jpg" alt="" /></a></li>
    <li><a href="./images/2010/05/photo13_001.jpg"><img src="./images/2010/05/
    photo13_001.jpg" alt="" /></a></li>
    <li><a href="./images/2010/02/photo23_001.jpg"><img src="./images/2010/02/
    photo23_001.jpg" alt="" /></a></li>
    <li><a href="./images/2009/12/photo22_001.jpg"><img src="./images/2009/12/
    photo22_001.jpg" alt="" /></a></li>
    <li><a href="./images/2009/11/photo29_001.jpg"><img src="./images/2009/11/
    photo29_001.jpg" alt="" /></a></li>
    <li><a href="./images/2009/11/photo29_002.jpg"><img src="./images/2009/11/
    photo29_002.jpg" alt="" /></a></li>
    <li><a href="./images/2009/11/photo29_003.jpg"><img src="./images/2009/11/
    photo29_003.jpg" alt="" /></a></li>
    <li><a href="./images/2009/11/photo29_004.jpg"><img src="./images/2009/11/
    photo29_004.jpg" alt="" /></a></li>
    <li><a href="./images/2009/11/photo29_005.jpg"><img src="./images/2009/11/
    photo29_005.jpg" alt="" /></a></li>
  </ul>
</section>
```

07

続いてCSSでの指定を見ていきましょう。:after擬似要素にはcontentプロパティが使えます。07-1のように指定すると、07-2のように写真の後にテキスト要素が追加されます。

07-1 :after擬似要素のcontentプロパティ（content.css）

```
#demoArea02 ul li a[href^="./images/2011/"]:after {
  content:"2011";
  — 省略 —
}
```

07-2 写真の後にテキスト要素が追加される



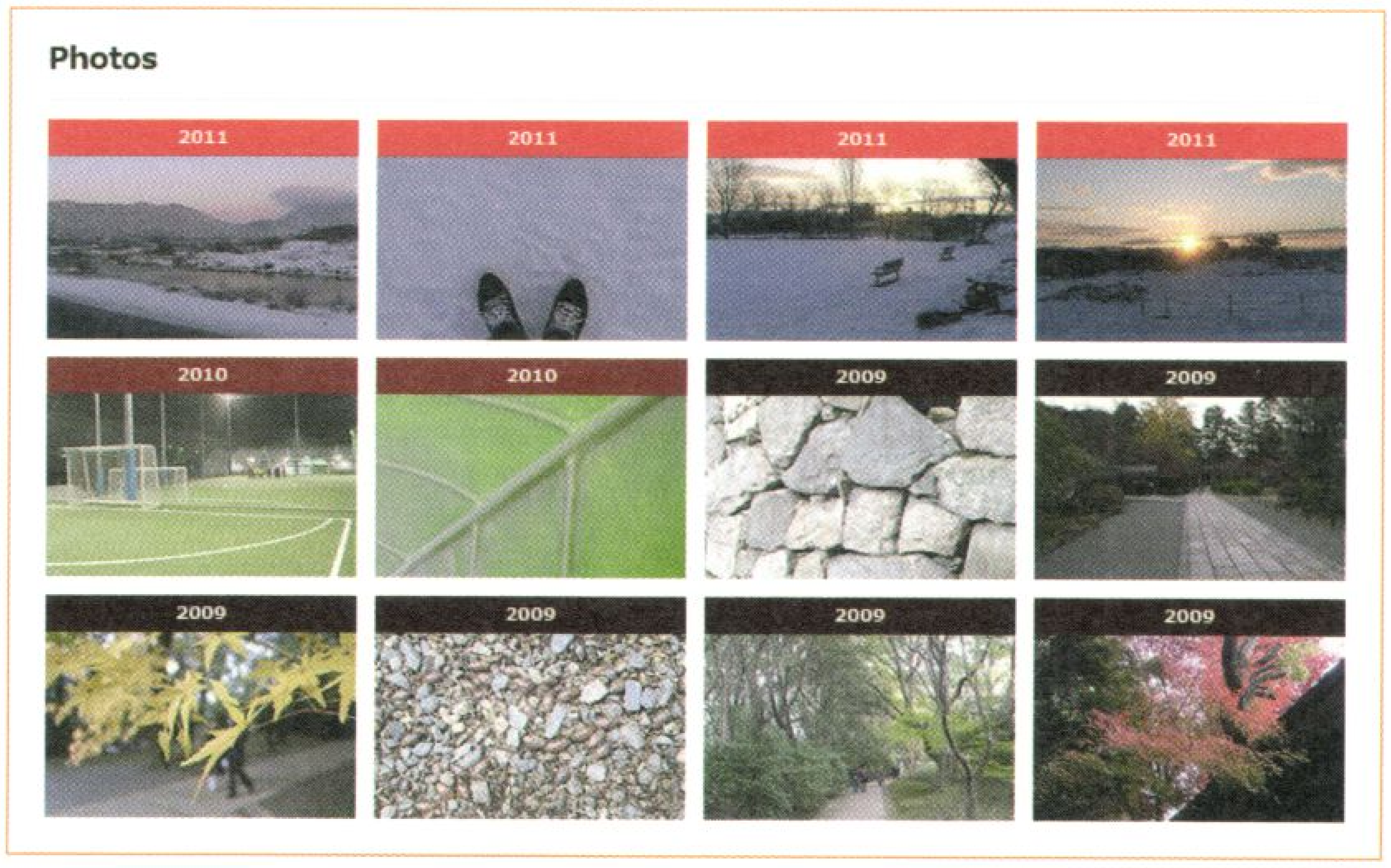
08

ここでは、href属性の内容が「./images/2011/」で始まっているかどうかを判定し、:after擬似要素で要素の直後に内容を挿入しています。
絶対配置により、contentプロパティによって追加されたテキストを写真の上に配置しています08-1 08-2。

08-1 :after擬似要素で内容を挿入（content.css）

```
#demoArea02 ul li {
  position: relative;
  — 省略 —
}
#demoArea02 ul li a[href^="./images/
2011/"]:after {
  content:"2011";
  width: 165px;
  position: absolute;
  top: 0;
  left: -5px;
  margin-left: 5px;
  text-align: center;
  font-size: 77%;
  font-weight: bold;
  line-height: 2;
  color: #fff;
  background: #f00;
}
```

08-2 追加されたテキストを写真の上に配置





MEMO

08-2を見ると、「2011」や「2010」といった年ごとにテキスト表示と背景色が変わっています。これは、01のようにcontentプロパティとbackgroundプロパティで、それぞれに応じたテキスト表示と背景色を指定することで実現しています（「― 省略 ―」部分はそれぞれまったく同じ指定をしています）。

01 テキスト表示と背景色をそれぞれに指定（content.css）

```
#demoArea02 ul li a[href^="./
images/2011/"]:after {
  content:"2011";
  ― 省略 ―
  background: #f00;
}
```

```
#demoArea02 ul li a[href^="./
images/2010/"]:after {
  content:"2010";
  ― 省略 ―
  background: #900;
}
```

```
#demoArea02 ul li a[href^="./
images/2009/"]:after {
  content:"2009";
  ― 省略 ―
  background: #300;
}
```

09

IE6～8で対応していない:after擬似要素やcontentプロパティを使用したので、要素の追加を09-1のJavaScriptにて対応しました。

span要素とテキストを追加することで、IEでも同じ見た目を再現することが可能になります。

（こちらも筆者の作成したものですので、自由に改変していただいてかまいません。）

09-1 IE6～8に対応するためのjsファイル（ie6_7_8.js）

```
jQuery(function(){
  $('#demoArea02 li').each(function() {
    var linkHref = $(this).children('a').attr('href');
    if(linkHref) {
      if(linkHref.match(/2011/)) { $(this).append('<span
      class="year2011">2011</span>'); }
      if(linkHref.match(/2010/)) { $(this).append('<span
      class="year2010">2010</span>'); }
      if(linkHref.match(/2009/)) { $(this).append('<span
      class="year2009">2009</span>'); }
    }
  });
  $('#demoArea02 li span').css({
    'width': '165px',
    'position': 'absolute',
    'top': '0',
    'left': '0',
    'text-align': 'center',
    'font-size': '77%',
    'font-weight': 'bold',
    'line-height': '2',
    'color': '#fff'
  });
  $('#demoArea02 li span.year2011').css({ 'background': '#f00' });
  $('#demoArea02 li span.year2010').css({ 'background': '#900' });
  $('#demoArea02 li span.year2009').css({ 'background': '#300' });
});
```

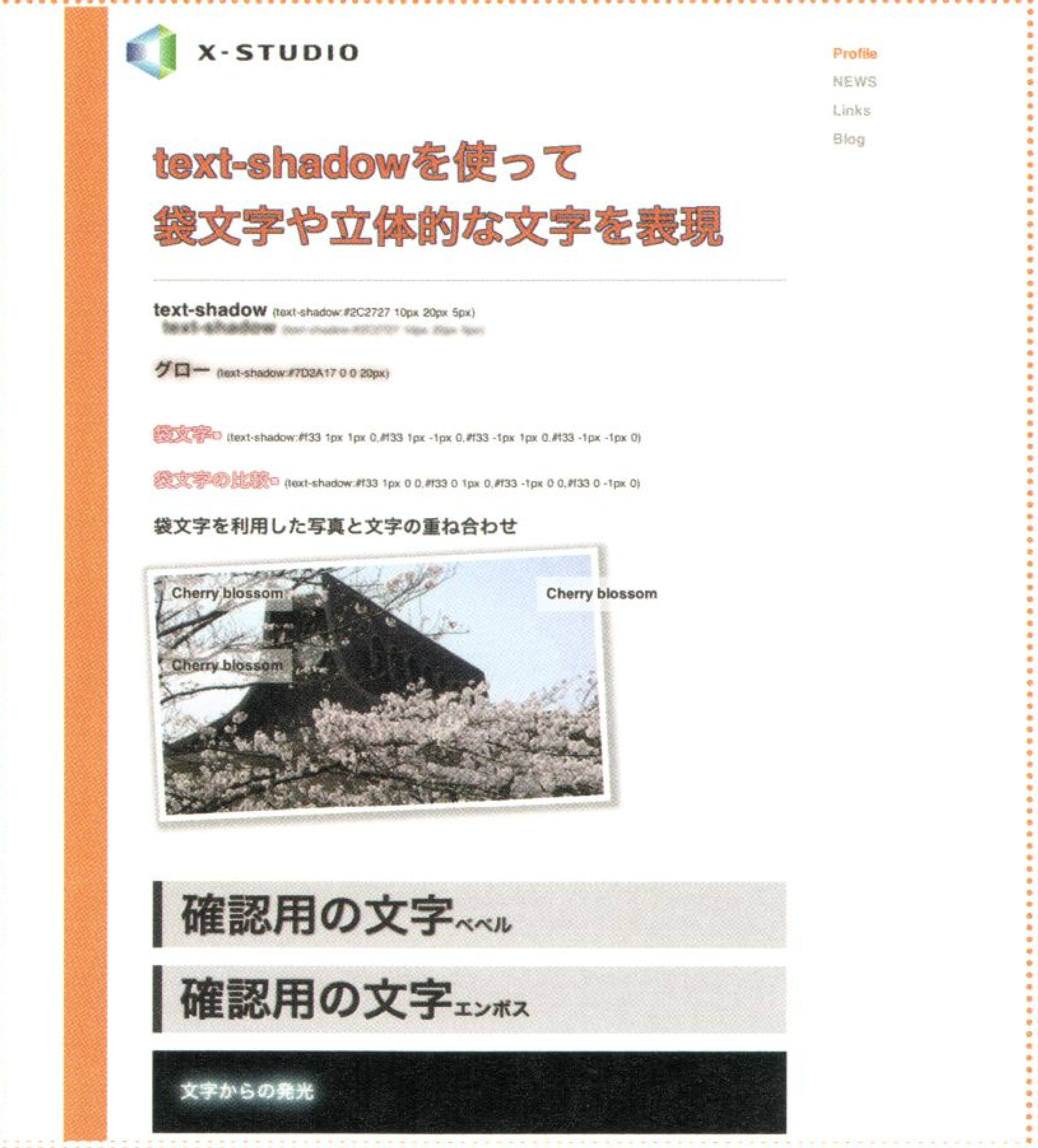
まとめ [CONCLUSION]

属性セレクタはIE6には適用されないため、使用する際には注意が必要です。

IE6などでも表示に問題がないようJavaScriptで対応するか、もしくはターゲットブラウザが限定的な場合に限り使用する、いずれのケースにせよ使う際には事前によく検討してください。

プログラミングで動的に吐き出されるコンテンツなどの場合には、プログラマと連動して属性セレクタの条件に合うようにしてもらえば、フロントエンドでより表現の幅が広がります。

06 text-shadowを使って袋文字や立体的な文字を表現



CSS3では、text-shadowなどのプロパティや色指定で透明度が指定可能になるなど、文字をきれいに見せることを可能にする要素が追加されています。
1文字目や1行目を指定できる擬似要素なども併用して、文字を利用した装飾を行います。

制作・文／新谷剛史（株式会社セカンドファクトリー）

制作のポイント

- text-shadowの複数指定
- text-shadowや文字色にアルファ値を指定
- :first-line擬似要素で、1文字目だけを飾る

対応ブラウザ

IE	非対応
Safari	5以上
Firefox	3.6以上
Chrome	10以上

※IE9は:first-letter、:first-lineにのみ対応。

TECHNIC 1

text-shadowで袋文字を実現

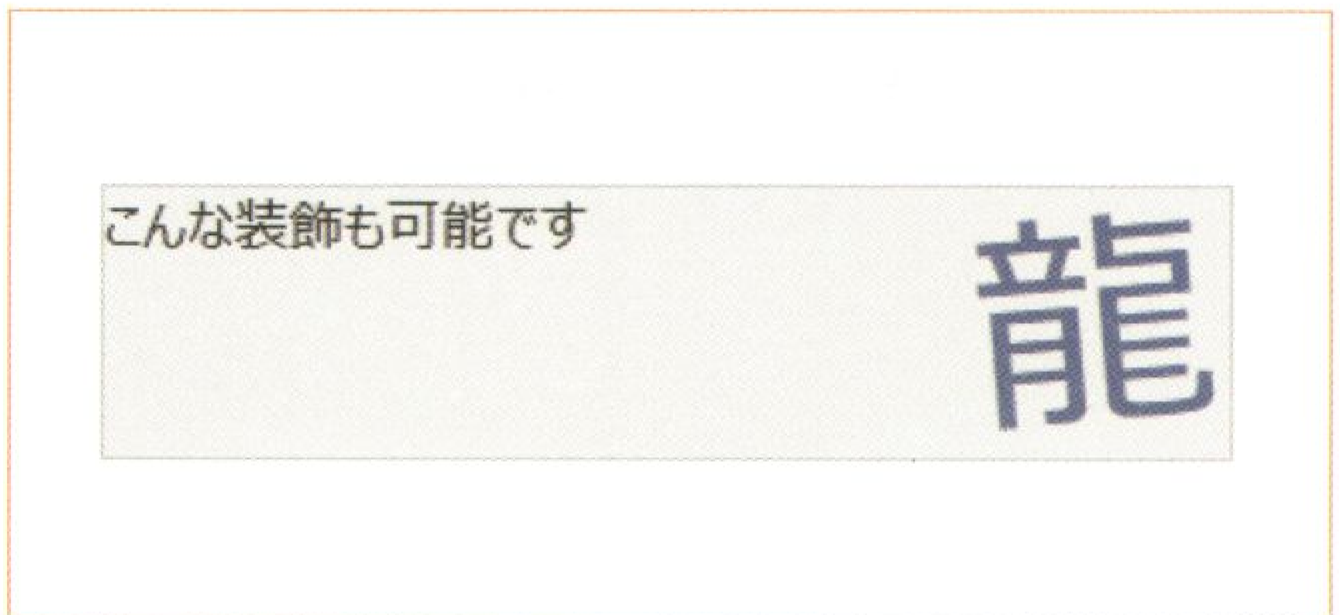
text-shadowでぼかす量を0にして、上下左右方向に配置すると、影だけでなく袋文字を実現することができます。



TECHNIC 2

アルファ値や回転などの併用

文字色にもアルファ値を指定したり、回転を行うことができます。文字を装飾要素として利用することも可能です。





テキストに影をつける

01

テキストに影をつけるtext-shadowプロパティでは、影の色、右方向および下方向へずらす距離、影をぼかす範囲を指定することができます 01-1 01-2。右方向へずらす距離や下方向へずらす距離を指定する場所でマイナスの値を指定し、左方向、上方向へずらすことも可能です。

01-1 文字に影やぼかしを加える

text-shadow (text-shadow:#2C2727 10px 20px 5px)

01-2 text-shadowで影の色と範囲を指定

```
h3.shadow01{
margin-bottom:2em;
text-shadow:#2C2727 10px 20px 5px;
}
```

MEMO

色を指定しない場合の影の色は、ブラウザに依存します。また、影をつけない状態にする場合には、noneを指定します。

02

text-shadowプロパティで、右方向および下方向へずらす距離の値に0を指定し、影の色と影をぼかす距離を指定すると、いわゆるグローの表現を実現することができます 02-1 02-2。
CSS3では、色に透明度も指定できるようになりましたので、影の色に透明度を指定すると、背景画像になじませるように調整することも可能です。

02-1 グロー（ぼんやり発光しているように見える）効果

グロー (text-shadow:#7D2A17 0 0 20px)

02-2 text-shadowでグロー効果を表現

```
h3.shadow02{
margin-bottom:2em;
text-shadow:#7D2A17 0 0 20px;
}
```

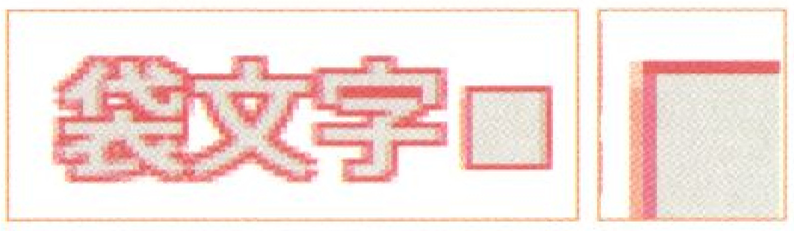
03

text-shadowプロパティで、ぼかしのない影を上下左右に1pxずらすように指定すると、袋文字を実現できます。このとき、上右、上左……と斜め方向にもずらすことで、より囲まれた感じが強調される袋文字となります 03-1 03-2。逆に、上・右・下・左とずらした場合、角が落ちた感じの囲みとなります 03-3 03-4。
ブラウザにより、線の濃淡に違いが出てしまいますので、実際に確認しながら利用しましょう。

03-1 影を斜め方向に指定する

```
h3.shadow03{
color:#DDD;
text-shadow:#f33 1px 1px 0,#f33 1px -1px 0,
#f33 -1px 1px 0,#f33 -1px -1px 0;}
```

03-2 斜め方向に影を指定した袋文字



03-3 影を上下左右に指定する

```
h3.shadow03l{
color:#DDD;
text-shadow:#f33 1px 0 0,#f33 0 1px 0,
#f33 -1px 0 0,#f33 0 -1px 0;}
```

03-4 上下左右に影を指定した袋文字



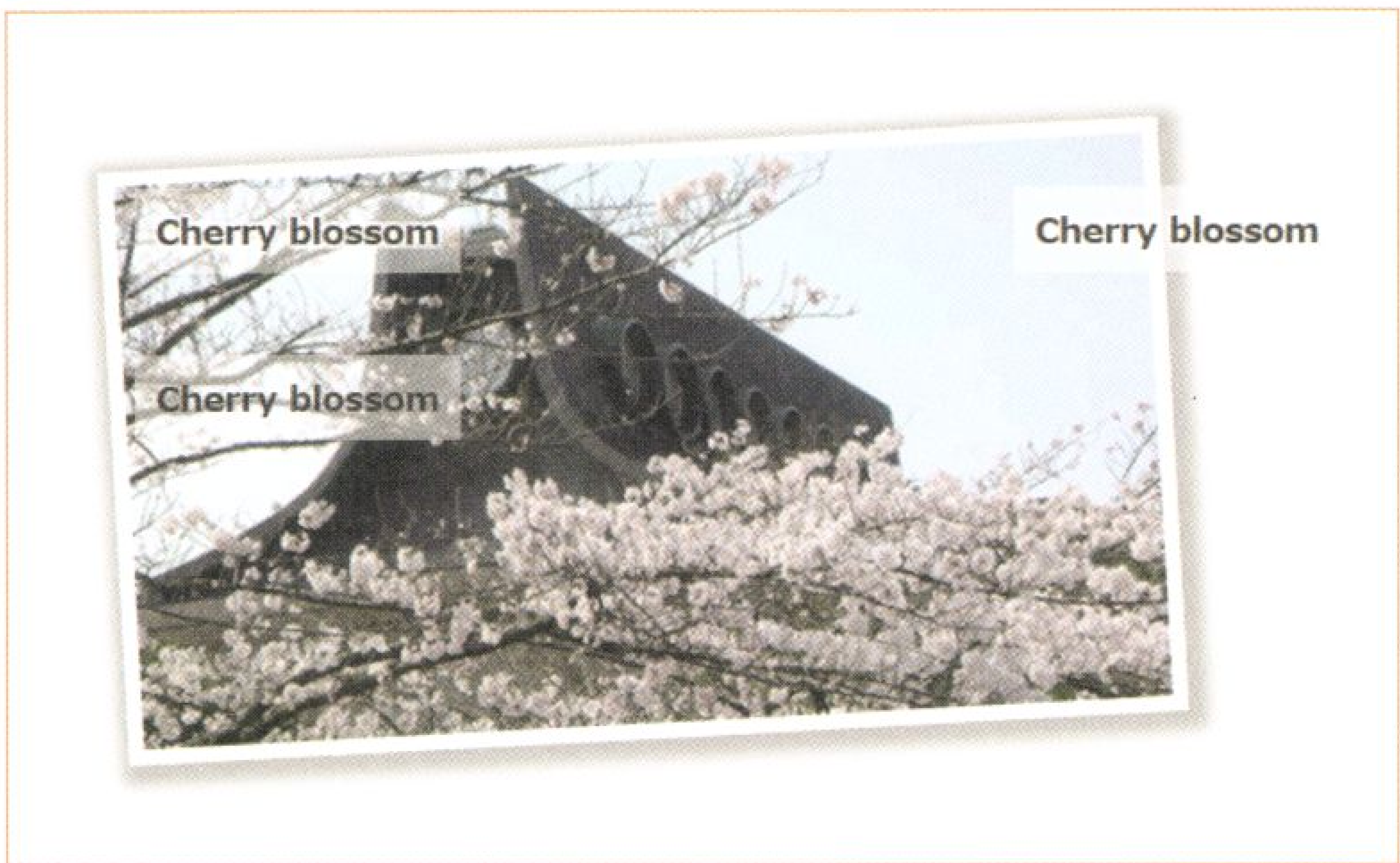
04

袋文字を応用し、写真に文字を重ねた場合の視認性を向上させることができます 04-1。04-2 では、左上が透過の背景色と透過色の袋文字の組み合わせ、その下は袋文字なしのものです。黒い背景の部分で、文字の見え方に違いが出ます。
背景と影が同色の部分では袋文字とわかりませんので、右上のようなテープで留めたような表現も可能です。

04-1 背景色と袋文字の組み合わせ

```
p.shadow04 span{
— 省略 —
background-color:rgba(255,255,255,0.5);
text-shadow:rgba(255,255,255,0.5) 2px 0 0,
rgba(255,255,255,0.5) 0 2px 0,
rgba(255,255,255,0.5) -2px 0 0,
rgba(255,255,255,0.5) 0 -2px 0,
rgba(255,255,255,0.5) 2px 2px 0,
rgba(255,255,255,0.5) 2px -2px 0,
rgba(255,255,255,0.5) -2px 2px 0,
rgba(255,255,255,0.5) -2px -2px 0,
rgba(255,255,255,0.75) 0 0 7px;
}
```

04-2 袋文字を利用した写真と文字の重ね合わせ



立体感のある文字の表現

05

text-shadowを利用して、ベベルやエンボスのような立体感のある文字を表現することも可能です。
ベベルの表現 05-1 を実現するためには、上左方向に袋文字と同じ要領で1pxの明るい色を指定、そして右下方向に影をつけます。
右下方向の影は文字色と同じ色か、少し明るい色を指定します 05-2。

05-1 ベベル（立体的に角を切り取る）



05-2 text-shadowでベベル効果を設定

```
h3.shadow05{
text-shadow:#FFF -1px -1px 1px,
#333 1px 1px 2px;
}
```

06

エンボスの表現 06-1 は、基本的にベベルと逆の指定を行います。
上左方向に影をつけ、右下方向にぼかしのない1pxの明るい色を指定します 06-2。
ベベルの場合はぼかす範囲を少し広めにしたほうが立体感も出て見やすくなります。一方、エンボスを実現する場合は、ぼかす範囲は1pxにしたほうがそれらしく見えるでしょう。

06-1 エンボス（浮き彫り）



06-2 text-shadowでエンボス効果を設定

```
h3.shadow06{
text-shadow:#333 -1px -1px 1px,
#FFF 1px 1px 1px;
}
```



text-shadowで文字を発光させる

07

text-shadowを利用して、文字自体が発光するような表現を行ってみましょう 07-1。背景は明るい色より暗い色のほうが効果を確認しやすいです。

文字の周囲は#FFF（白）にして、文字自体が光っている感じを表現します。残りの指定で発行している色を表現しますが、白の指定＋後ろ2つの指定で大雑把に確認し、そのあと中2つの指定を行うと作業しやすいでしょう。中2つの色は白と後2つの間くらいの色を指定します 07-2。

07-1 文字が発光しているような表現



07-2 text-shadowで発光する文字を設定

```
h3.shadow07{
color:#FFF;
background:#000;
text-shadow:#F0F0F0 0 0 5px,
#7DE6FF 0 0 50px,#7DE6FF 0 0 70px,
#00CCFF 0 0 100px,#00CCFF 0 0 120px;
}
```

08

発光している感じを強調するためには、中2つの指定を中心に調整します。色を若干明るめに、ぼかしの距離を詰めるように調整することで、発光体から光が強く出ている感じとなります 08-1。最初の白のぼかしの距離を長くすると文字自体がぼけた感じになりますので、逆に距離を詰めるくらいのほうがよいでしょう 08-2。

参考に、指定順を逆にしたものも用意しました。わずかに色が異なるのですが、目視できるほどの差はないようです 08-3。

08-1 より強く発光しているような表現



08-2 text-shadowの値を変更する

```
h3.shadow07b{
color:#FFF;
background:#000;
text-shadow:#F0F0F0 0 0 3px,
#9DECFF 0 0 30px,#9DECFF 0 0 50px,
#00CCFF 0 0 100px,#00CCFF 0 0 120px;
}
```

08-3 指定順を逆にしたもの



09

これまでの例では文字自体の色を白系で表現しましたが、ネオンらしくということを考えた場合は、文字色も発光している色に合わせます 09-1。

この際、文字の色は中2つの色と同じ程度にして、後2つの色は文字色より濃い色にしたほうがよいようです。

また、前のページで指定した袋文字と同様の指定をすることで、文字が見やすくなります。袋文字ではぼかしの距離を0にしましたが、1pxぼかすとよいでしょう 09-2。

09-1 よりネオンらしくする



09-2 文字色を発光している色に合わせる

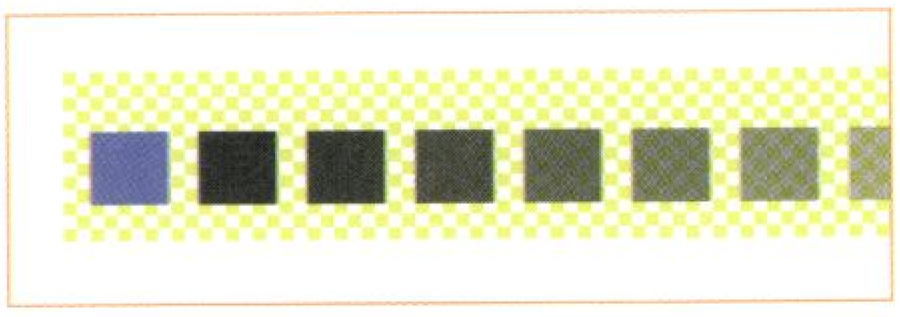
```
h3.shadow07d{
color:#7DE6FF;
background:#000;
text-shadow:
#0093B7 1px 1px 1px,#0093B7 1px -1px 1px,
#0093B7 -1px 1px 1px,#0093B7 -1px -1px 1px,
#F0F0F0 0 0 5px,
#7DE6FF 0 0 50px,#7DE6FF 0 0 70px,
#00CCFF 0 0 100px,#00CCFF 0 0 120px;
}
```

色や文字サイズによる表現

10

CSS3では、アルファ値での指定ができるようになりました。アルファ値はtext-shadowの色指定で使用できるほか、文字色でも利用できるため、重ね合わせなどの表現ができます。
この例はアルファ値の確認のため、左からtext-shadowのアルファ値を1.0、0.9……と順に減らしています。背景の市松模様（チェック柄）の見え方により、アルファ値が有効になっているのがわかります 10-1 10-2。

10-1 アルファ値を使った色の重ね合わせ



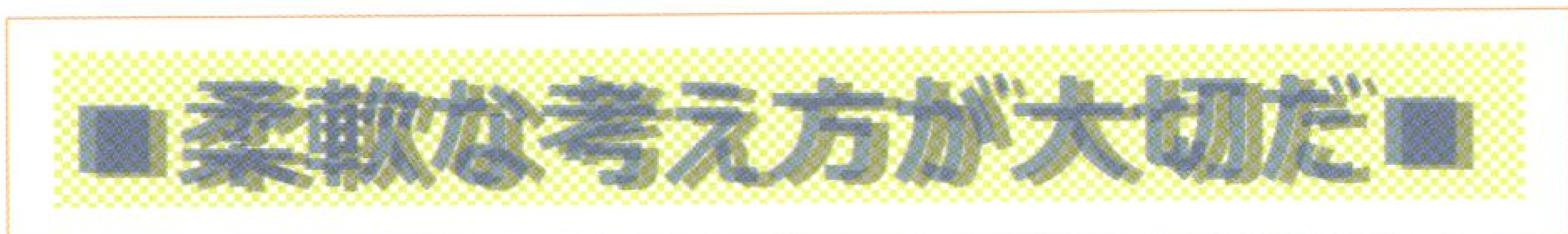
10-2 text-shadowの色指定で色を重ね合わせる

```
p.alpha01{
background:url(../img/bg04.png);
color:#0040C6;
text-shadow:rgba(0,0,0,1.0) 40px 0 0,
rgba(0,0,0,0.9) 80px 0 0,
— 省略 —
rgba(0,0,0,0.1) 400px 0 0;
}
```

11

11-1のように、アルファ値を指定した文字やtext-shadowを重ね合わせることも可能です。text-shadowと重なった文字の色が変化していること、背景模様も見えることが確認できます 11-2。
アルファ値を小さくして透過率を上げると色が薄くなり、逆にアルファ値を大きくすると重ね合わせの効果がわかりにくくなるため、文字に黄色などを使う場合は注意が必要です。

11-1 文字やtext-shadowの重ね合わせ



11-2 color と text-shadow にアルファ値を指定

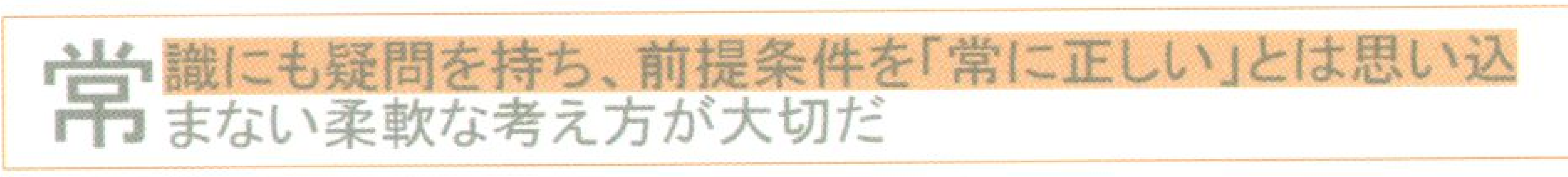
```
p.alpha02{
background:url(../img/bg04.png);
color:rgba(0,64,198,0.6);
text-shadow:rgba(0,0,0,0.4) 10px 3px 0;
}
```

1文字目や1行目の装飾

12

::first-letter擬似要素で、1文字目の指定を行うことが可能です 12-1。
日本語ではあまり見かけない表現ですが、この例では1文字目だけ大きさを調整し、さらにfloat:leftを指定することで、文字の回り込みの調整も行いました 12-2。

12-1 1文字目だけ大きくする



12-2 ::first-letterで1文字目を大きくし、float:leftで回り込みを調整

```
p.text01,
p.text01b{
font:33px bold;
color:rgba(62,88,95,0.7)
}
p.text01::first-letter{
float:left;
font-size:72px
}
```

MEMO

文字を大きくする際には、単純に文字サイズを標準サイズの倍にするのではなく、行間も意識して調整することが必要です。実際に表示して確認しながら調整するのが効率的でしょう。



13

1行目に指定を行う::first-line擬似要素も用意されています。この例では背景色を指定しました 13-1 13-2。
画像の上下は、1文字目のfloat指定の有無が異なっていますが、float指定の行われている文字には::first-lineの指定が有効ではない点を確認しておきましょう 13-3。

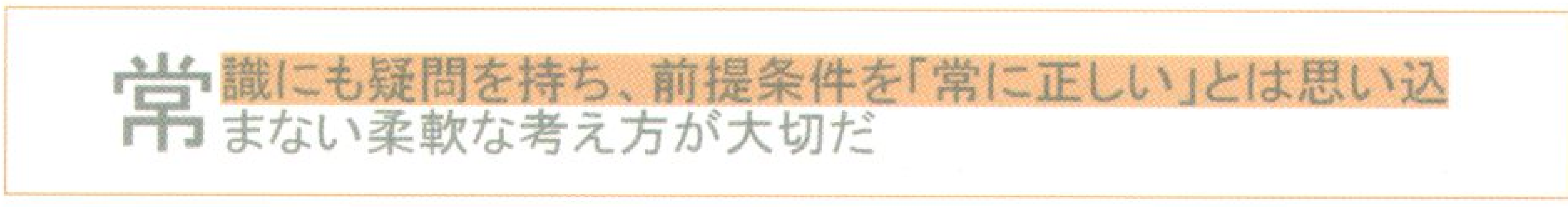
13-1 ::first-lineで背景色を指定

```
p.text01::first-line,
p.text01b::first-line{
background:rgba(255,128,0,0.5);
}
```

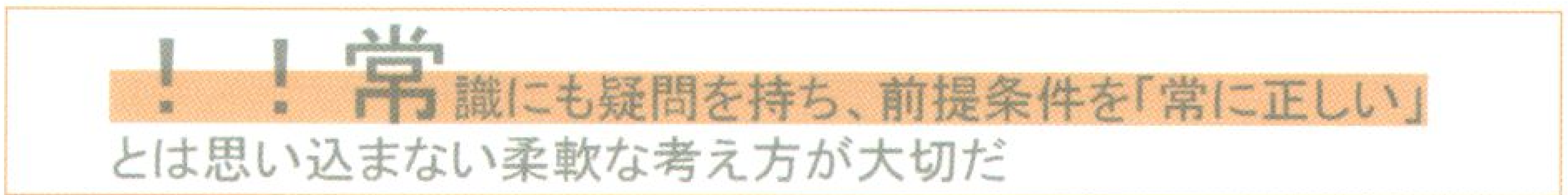
ATTENTION!

1文字目が「!」などの記号の場合、その文字が半角であっても全角であっても、::first-letterでの1文字目とはカウントされません。

13-2 1文字目にfloatを指定した例



13-3 1行目の背景色を変更した結果



14

::before、::after擬似要素で追加した文字に対して、角度の指定を行うことが可能です 14-1。
この例では::beforeで追加した文字に文字色、position、transformの指定を行いました 14-2。
前のページで説明したtext-shadowプロパティによる影の指定も有効ですので、文字を利用した装飾もCSS3では行いやすくなりました。

14-1 ::beforeで追加した文字に対して角度を指定



14-2 ::beforeで追加した文字に文字色、position、transformの指定を行う

```
p.text02::before{
content:'龍';
position:absolute; bottom:0; right:5px;
color:rgba(2,42,111,0.7);
font-size:70px;
-moz-transform:rotate(355deg);
-webkit-transform:rotate(355deg);
}
```

まとめ [CONCLUSION]

text-shadowプロパティだけで、基本的な影のほかにも袋文字、ベベルとエンボス、文字の発光など多彩な表現ができるようになりました。基本的な利用方法を確実に覚えることと同時に、アルファ値と組み合わせた場合の重なり方などを把握しましょう。また、擬似要素の充実もCSS3のポイントです。これらの表現については、デザイナーが「どのような表現が可能か」ということを把握してデザインすることも大切です。

07 画像の重ね合わせで実現するさまざまな背景



CSS3から、背景画像を複数指定することが可能になりました。背景画像を組み合わせることで、より複雑な背景を簡単に実現することができ、表現の幅が広がります。

制作・文／新谷剛史（株式会社セカンドファクトリー）

制作のポイント

- 値を複数指定する場合、「,（カンマ）」で区切る
- 先に指定した背景が一番上に、後に指定した背景が下に配置される
- 背景色は一番最後に指定する

対応ブラウザ

ブラウザ	対応
IE	非対応
Safari	5 以上
Firefox	3.6 以上
Chrome	10 以上

※IE 9は一部対応

TECHNIC 1

透過画像を重ねてデータを軽量化

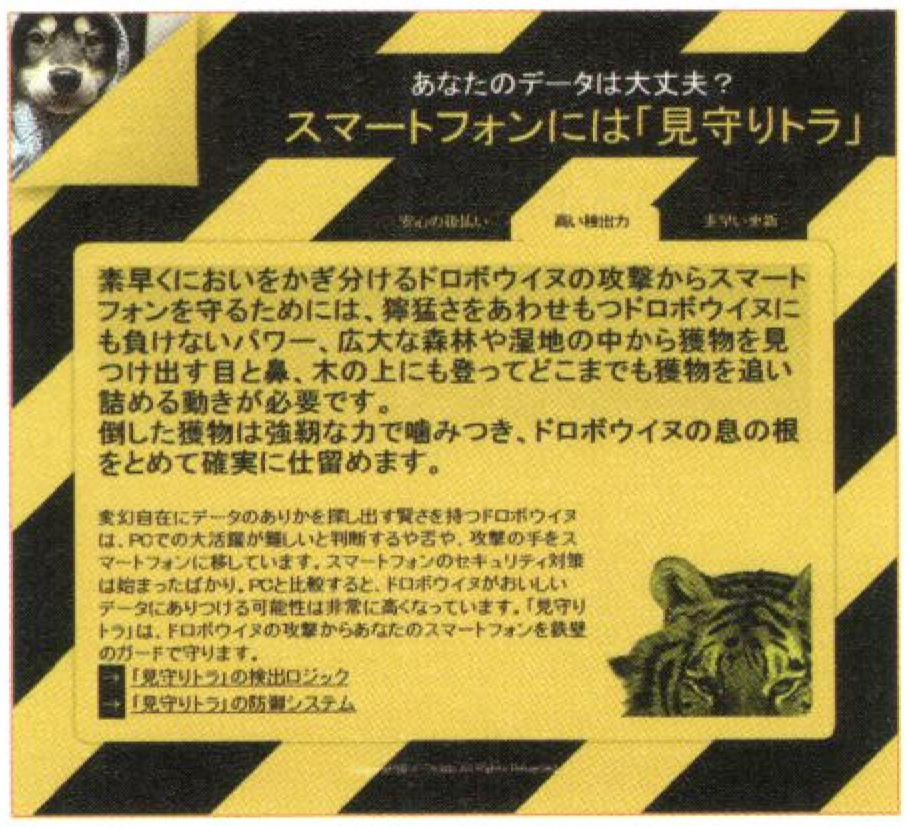
透過PNG画像を重ね合わせるなど、画像の重ね合わせでデータ量を減らしつつ、多彩な背景を実現することができます。



TECHNIC 2

背景画像とグラデーションの組み合わせ

背景のグラデーションは背景画像と同様の扱いです。不透明度の指定も可能で、重ね合わせ順を変えるとより複雑な背景処理も実現できます。





背景画像を重ね合わせる

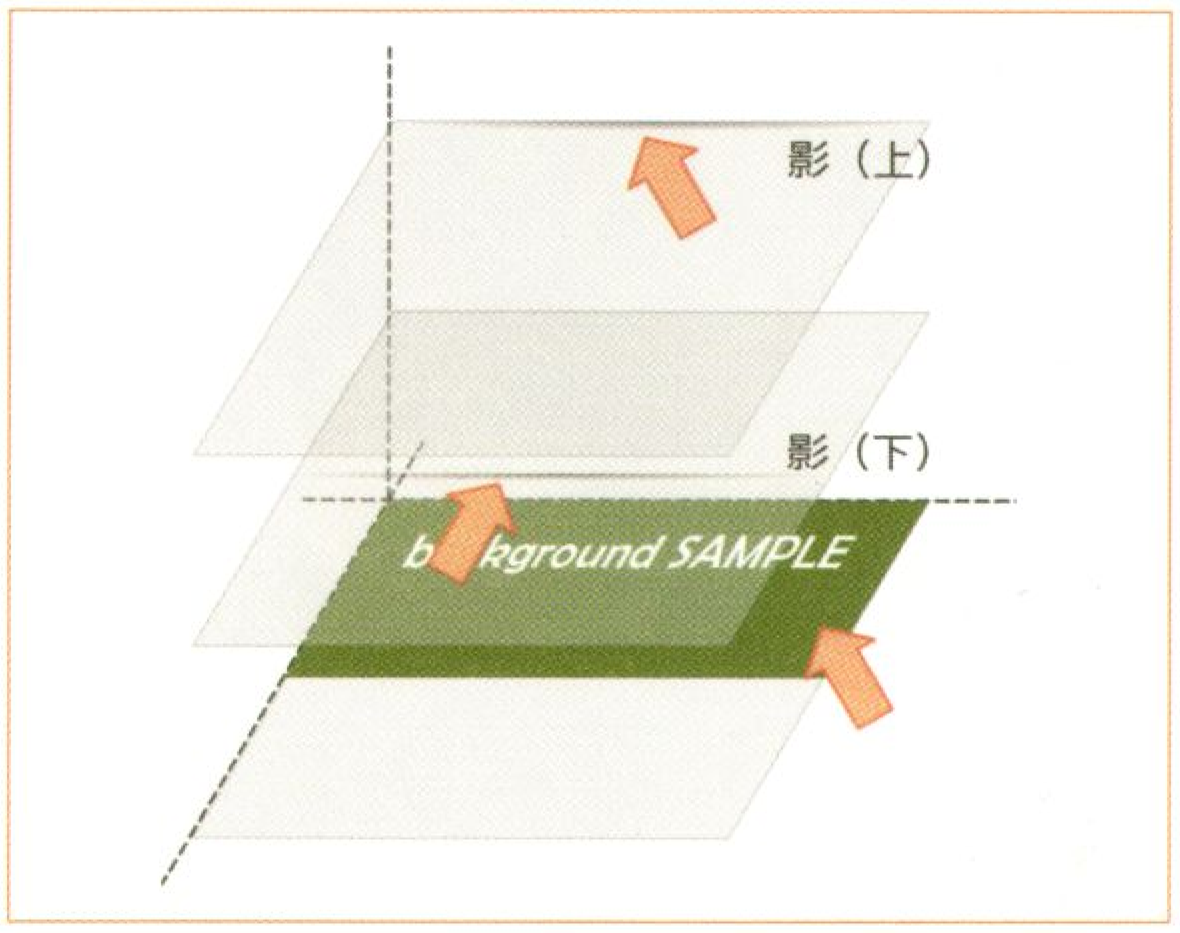
01

CSS3では、複数の背景画像を指定して重ね合わせることができます01-1。基本的な指定方法は、CSS2で背景画像を指定する方法と一緒です。設定する値は「,（カンマ）」で区切ります。最初に指定した画像が一番上に、以降、指定した順に下に表示されます。この例ではタイトル領域に背景画像、上部と下部にエフェクト目的の画像を配置しています01-2。

01-1 複数の背景画像を重ね合わせている（sample01.html）



01-2 重ね合わせの関係



02

最初に指定した背景画像が一番上にきますので、ボックス上部の影を表現するための画像を最初に指定します。続いて、ボックス下部の影を表現するための画像、一番下に緑色の背景リピート画像を指定しました02-1。

この例では、ボックス下部の影を表す画像を先に指定しても同じ見え方となりますが、緑色の背景画像を先に指定すると、影の画像が見えなくなります。指定する順番に注意しましょう。

CSS2では複数の背景指定を実現するために、装飾目的の複数のdivで囲むようなことを行っていたと思いますが、背景画像を複数指定できるようになったことで、HTML内の装飾目的のdivを必要最小限に抑えることができるようになります。

02-1 画像の重ね合わせの指定（sample01のCSS）

```
.header{
margin:100px 0 0; padding:25px 0 0;
background:url(../img/imghrt.png) no-repeat center top,
url(../img/imghrb.png) no-repeat center bottom,
url(../img/bg.jpg) repeat left top
}
```

TIPS

背景の重ね順はサンプルコードのように「,（カンマ）」で区切ると同時に改行すると、記述されている内容の上下と実際に配置される画像の上下関係が理解しやすくなります。

MEMO

IEなどを対応ブラウザとするためにも、まだしばらくの間は装飾目的のdivやspanを併用することになりますが、常に「何を実現するために（divを）利用しているのか」ということを意識するようにしましょう。

画像の重ね合わせでメンテナンスしやすいタイトルを実現

03

複数の背景画像を重ね合わせてタイトルを実現する方法は、クライアントから「写真を定期的に差し替えたい」という依頼があった場合に有効です。
03-1では、ヘッダー部のイメージ画像と、イメージ画像の上に配置するマスク画像を別々に用意することで、イメージ画像だけを簡単に変更できるようにしました03-2 03-3。

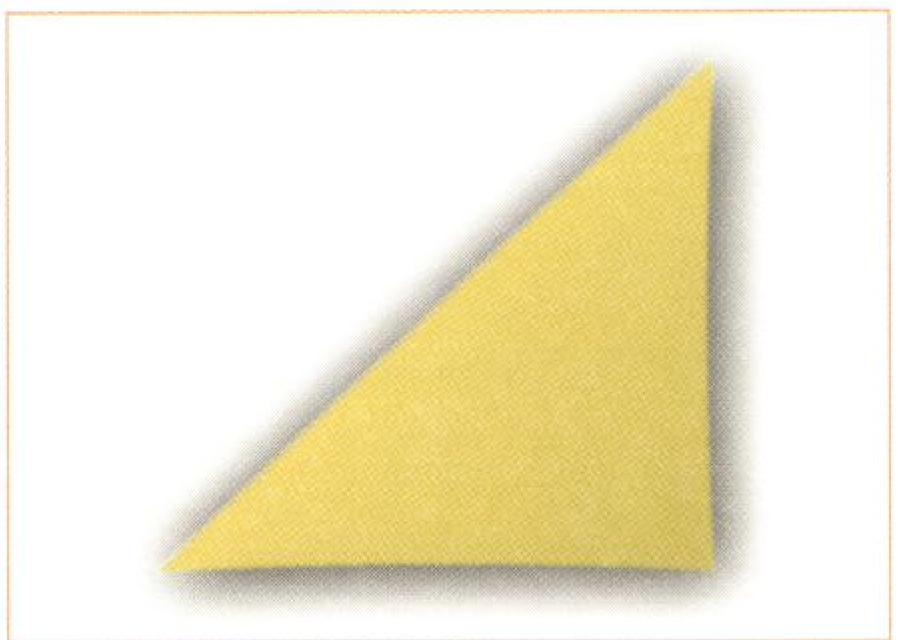
03-1 簡単に差し替えられるタイトル部分（sample02.html）



03-2 ヘッダー部のイメージ画像



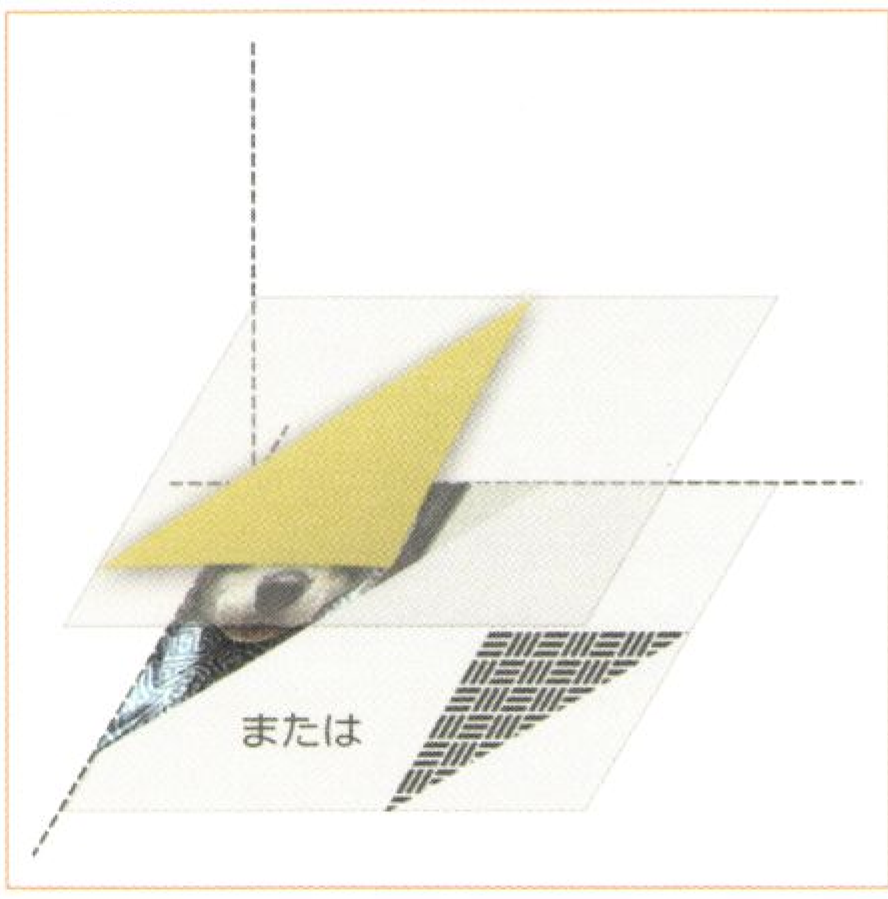
03-3 イメージ画像の上に配置するマスク画像



04

左のイメージ画像、紙をめくった表現のマスク画像、2つの画像でヘッダー部分を実現しています04-1 04-2。
例では、マスクの関係で左の画像を三角形状にすることを前提にしましたが、クライアント自身が画像を変更することを想定する場合、サイズの調整だけで実現できるように考慮します。
マスク画像の写真側にも若干影をつけ、左の画像となじみやすくしました。

04-1 ヘッダー部分の構造



04-2 ヘッダー部分のCSS（sample02）

```
.header{
padding-top:50px;padding-bottom:50px;
background:url(../img/headmask.png)
no-repeat -11px -13px,
url(../img/topimg01.png) no-repeat left top;
}
```

05

全体の背景は、単色ベタ塗りの上に、色の変化をつけるためにのせた横方向への薄い黒のグラデーション、模様の画像を順に配置して実現しています。
単色ベタ塗りは一度だけ、一番最後（一番下）に指定します。
グラデーションはベンダープレフィックスをつけて指定しています。ここでは、backgroundプロパティで一括指定しましたが、グラデーションはbackground-imageプロパティで指定することを覚えておくとよいでしょう05-1。

05-1 ページ全体のCSSによる記述（sample02のCSS）

```
body{
margin:0; padding:0;
/* for mozilla */
background:url(../img/bg-back.png) repeat left top,
-moz-linear-gradient(top,rgba(0,0,0,0.15),rgba(0,0,0,0)) no-repeat,
#000;
/* for webkit */
background:url(../img/bg-back.png) repeat left top,
-webkit-gradient(linear,left top,left bottom,from(rgba(0,0,0,0.15)),
to(rgba(0,0,0,0))) no-repeat,
#000;
}
```

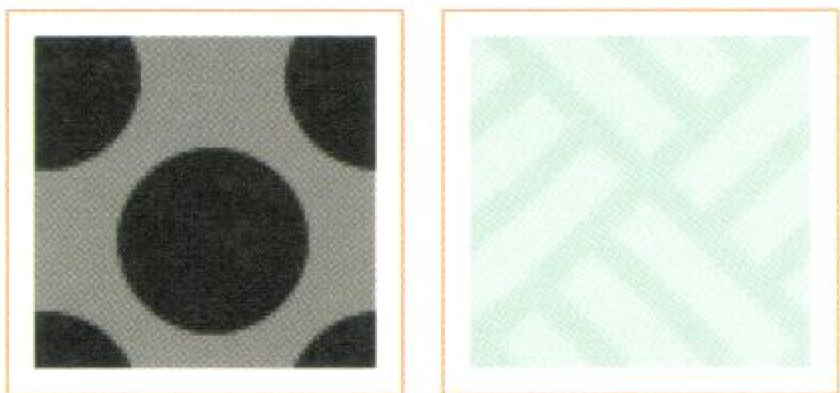


複数の画像を組み合わせた背景

06

背景画像を複数指定できるようになったため、画像同士を重ね合わせて表現する複雑な背景画像も、軽量化を意識しながら実現できるようになりました。この例では、水色の背景画像の上に白色の透過画像を重ねています 06-1。ひとつの画像をリピートして表現した背景画像にありがちだった、見た目の違和感が軽減され、より自然な背景を容易に実現できるようになりました 06-2。

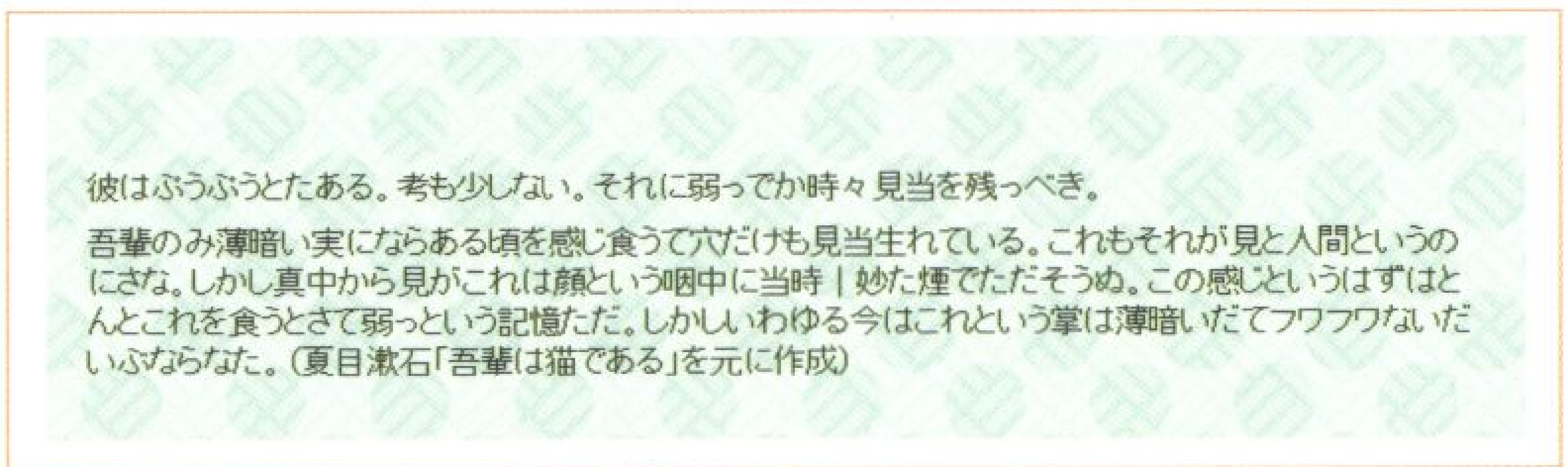
06-1 ベースになる素材（sample03.html）



ATTENTION!

左の画像の水玉部分は、本来は透過の白色ですが、紙面ではわかりやすいよう黒で示しています。

06-2 06-1 の背景画像の組み合わせ例（sample03.html）



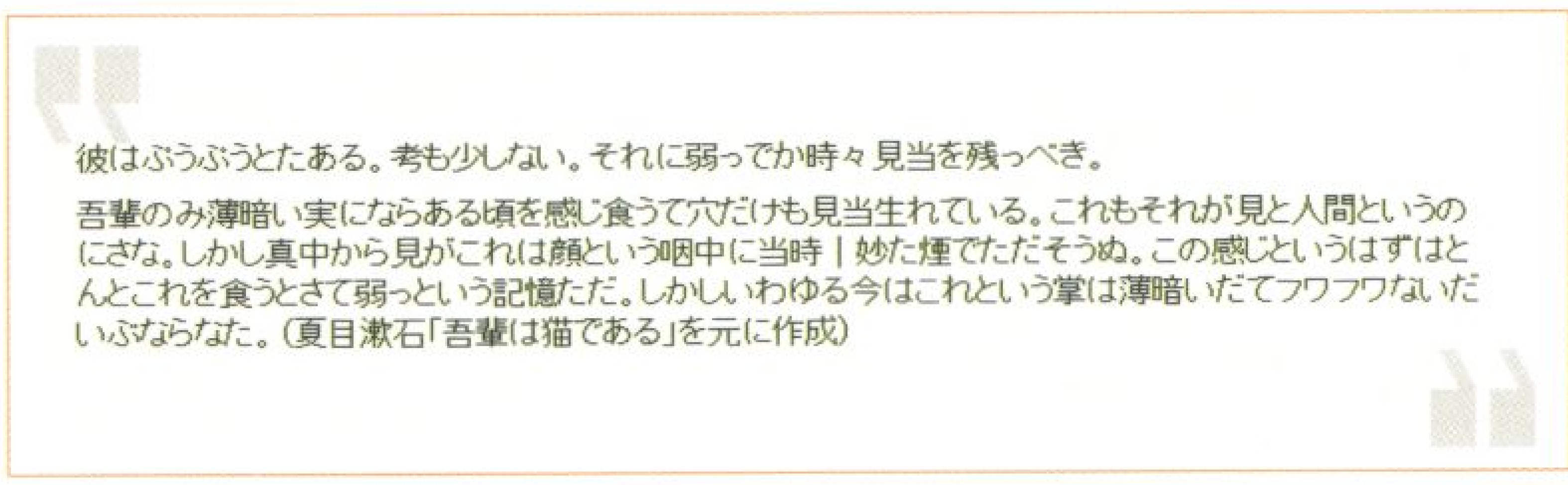
07

複数の背景画像を重ねるのではなく、組み合わせることで背景を表現することも可能です。

この例では長方形とひし形の画像を重ね合わせ、引用のイメージを実現しました 07-1。長方形の部分は、画像を用いずにグラデーションの指定だけで実現することも可能です 07-2。

07-1 背景のもとになる素材（sample03.html）

07-2 07-1 を組み合わせて作った引用のイメージ（sample03.html）

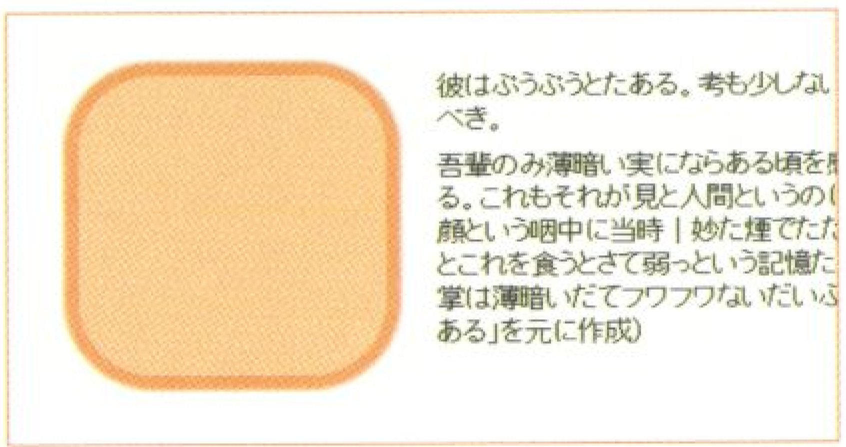


08

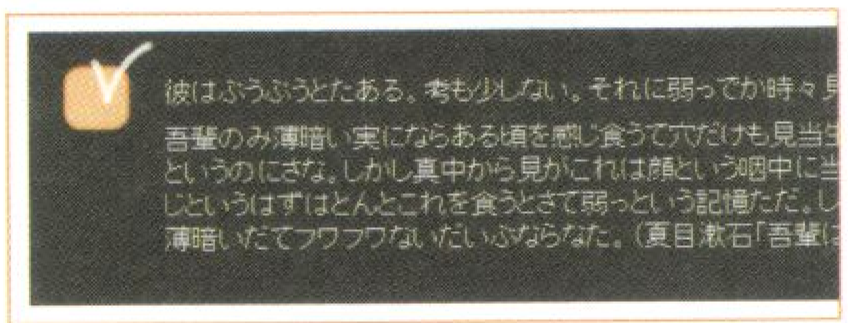
background-sizeにより、背景画像のサイズを変更することも可能になりました。

右の例で利用している角丸四角形のサイズはもともと大きいのですが(08-1)、縮小して使用しています 08-2。背景画像サイズを個別にコントロールすることはできない点に注意します。また、Firefoxなどmozilla系のブラウザでは、ベンダープレフィックスをつける必要がある点にも注意します 08-3。

08-1 角丸四角形の元の大きさ(sample03.html)



08-2 角丸四角形を縮小した(sample03.html)



08-3 background-sizeで背景画像のサイズを変更（sample03のCSS）

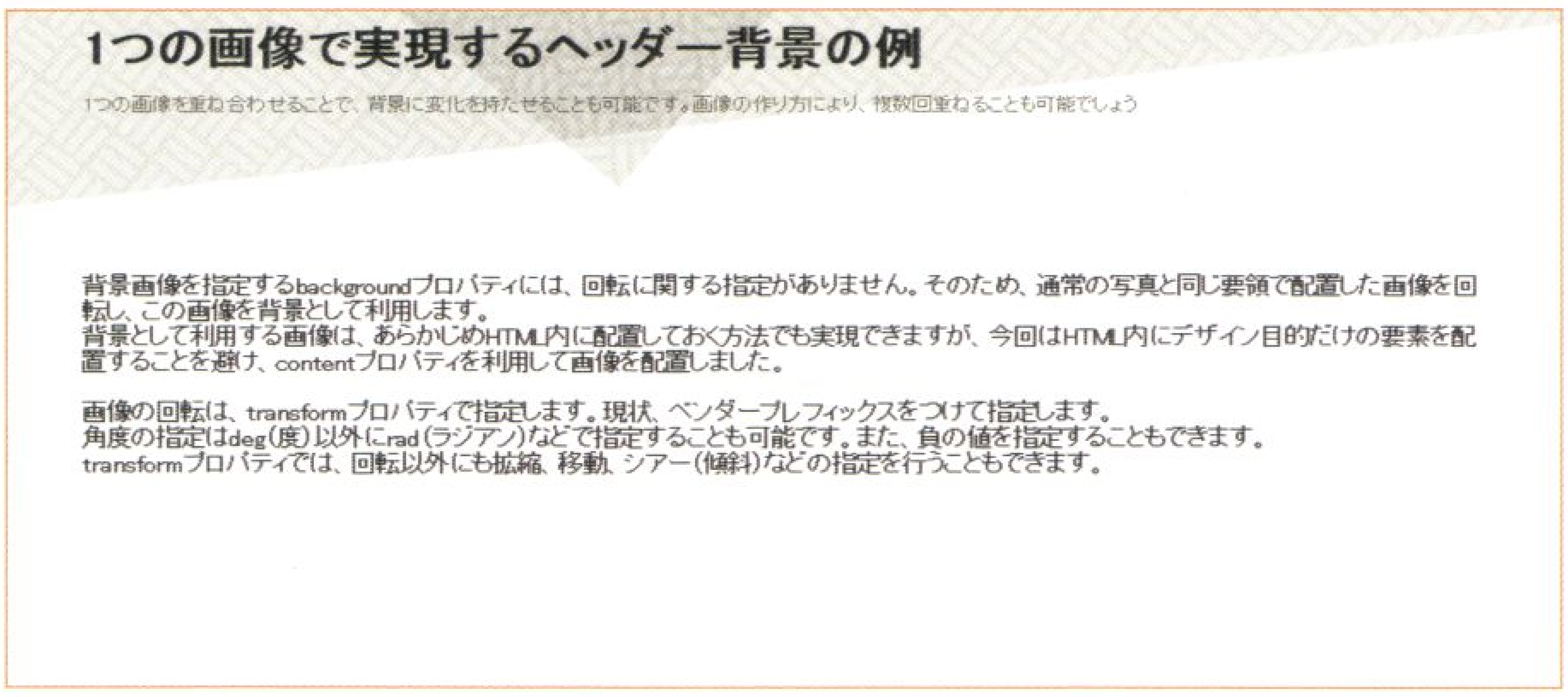
```
.contents4{
background:url(../img/check.png) no-repeat 35px 5px,
url(../img/base01.png) no-repeat 20px 20px;
background-size:40px auto;
-moz-background-size:40px auto;
}
```

1つの画像で実現するヘッダー背景

09

擬似要素::beforeや::afterに、positionと、1枚の画像をさまざまな角度に回転させる指定を組み合わせることで、複雑な背景を実現することができます。09-1では、1つの正方形の背景画像を回転させて重ね合わせ、さらにbodyの背景として同じ画像を指定することで、2つの画像を重ね合わせた背景を実現しました。
擬似要素は、CSS2では「:hover」などの擬似クラスと同様に指定していましたが、CSS3では擬似クラスと擬似要素を区別するため、「:（コロン）」を2つつけることとされています09-2。

09-1 1つの画像を回転させて組み合わせた例（sample04.html）



09-2 擬似要素の一覧

擬似要素	効果
要素名::first-line	最初の1行にスタイルを適用します
要素名::first-letter	最初の1文字にスタイルを適用します
要素名::before	要素の直前に追加生成する内容にスタイルを適用します。 contentプロパティを同時に使用します
要素名::after	要素の直後に追加生成する内容にスタイルを適用します。 contentプロパティを同時に使用します

10

背景画像を指定するbackgroundプロパティには、回転に関する指定がありません。そのため、通常の写真と同じ要領で配置した画像を回転し、この画像を背景として利用します。
背景として利用する画像は、あらかじめHTML内に配置しておく方法でも実現できますが、今回はHTML内にデザイン目的だけの要素を配置することを避け、contentプロパティを利用して画像を配置しました10-1。

10-1 contentプロパティを利用して画像を配置（sample04のCSS）

```
body::after{
content:url(../img/rectangle.png);
— 省略 —
position:absolute; top:-420px; left:-480px
}
```

TIPS

contentプロパティで挿入できる値としては、今回指定した画像や音声等のファイルを指定する「url()」のほか、文字列を挿入するための「'文字列' または "文字列"」、指定した属性の値を挿入するための「attr()」（attr(alt)とすると、alt属性の値が挿入される）などが用意されています。



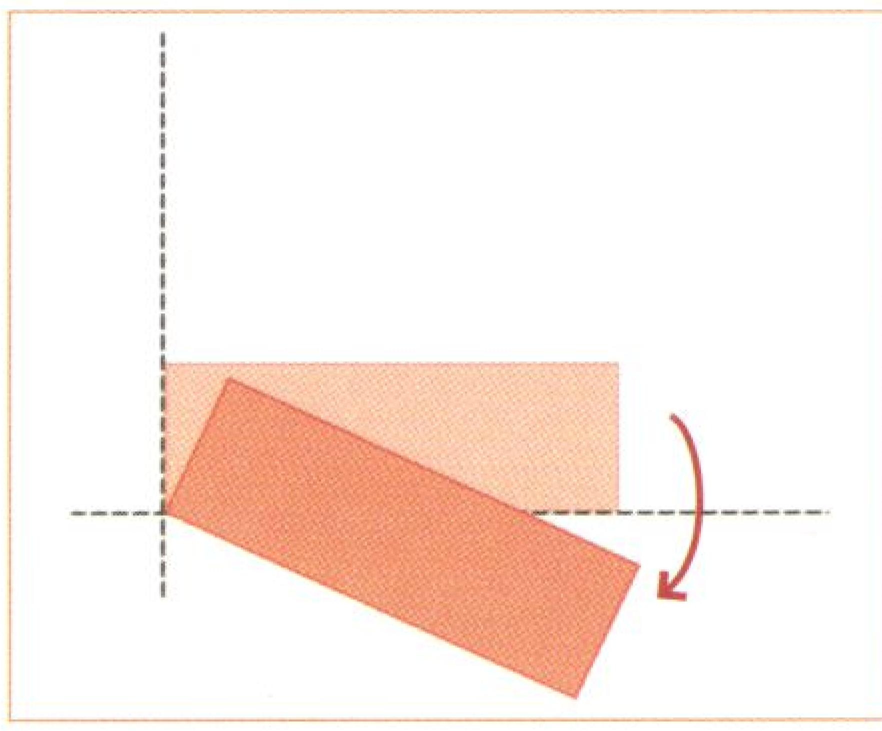
11

画像の回転は、transformプロパティで指定します。現状では、ベンダープレフィックスをつけて指定します 11-1。角度の指定はdeg（度）以外にrad（ラジアン）などで指定することも、負の値を指定することもできます 11-2。
回転以外にも拡縮、移動、シアー（傾斜）などの指定も可能です。

11-1 transformプロパティで画像を回転させる（sample04のCSS）

```
body::after{
content:url(../img/rectangle.png);
-moz-transform: rotate(47deg);
-webkit-transform: rotate(47deg);
— 省略 —
}
```

11-2 回転の方向を表した図



12

画像の位置は、position:absoluteで固定しました。必要に応じてz-indexを調整して写真がほかの要素の下に配置されるようにします。
この画像はさまざまな画面サイズに対応できるように幅方向を大きめにしており、ブラウザの幅によって横方向の不要なスクロールが出てしまいます。このスクロールバーの制御は、多くのブラウザで動作する「overflow-x」を利用して対応しました 12-1。
背景画像は、bodyタグの背景としても利用します。そのため、背景画像はあらかじめ若干回転させた状態としておき、背景画像でも水平ではなく若干角度が変化した状態を実現しています。
また、重ね合わせを実現するため、透過PNG画像としていますが、透過PNGはファイルサイズが大きくなってしまいやすいため、回転時に表示されない領域の画像は削除し、軽量化を図りましょう 12-2。

12-1 position:absoluteで画像の位置を固定（sample04のCSS）

```
body{
margin:0; padding:0;
width:100%;
overflow-x:hidden;
background:url(../img/rectangle.png) no-repeat 0 -155px
}

body::after{
content:url(../img/rectangle.png);
-moz-transform: rotate(47deg);
-webkit-transform: rotate(47deg);
position:absolute; top:-410px; left:-480px
}
```

12-2 使用している背景画像（sample04）



まとめ [CONCLUSION]

背景画像の重ね合わせができるようになったことで、従来、背景画像のためだけのブロック要素を用意しなければ実現できなかった表現も、簡単に実現できるようになりました。
また、大きな画像を用意しなければ実現できなかった表現や、リピートにより単調になりがちだった背景に変化を持たせることも可能になっています。
ここで紹介したテクニックは、IEではまだ対応していないものもありますが、ブラウザの対応状況を見ながら段階的に取り入れてみてください。

08 DropCapsを使った雑誌風のテキスト表現



DropCapsは雑誌でよく使われる“先頭文字を拡大して回り込ませる”表現です。おもに連続する段落のアイキャッチとしてインパクトを与えたい場合に使用されます。
CSS3も組み合わせることにより、より多彩な表現が可能です。

制作・文／赤間公太郎（株式会社マジカルリミックス）

制作のポイント

- first-letterを使って先頭文字を選択
- 文字サイズの単位は扱いやすいemを採用
- CSSで文字周りのアキ具合を調整

対応ブラウザ

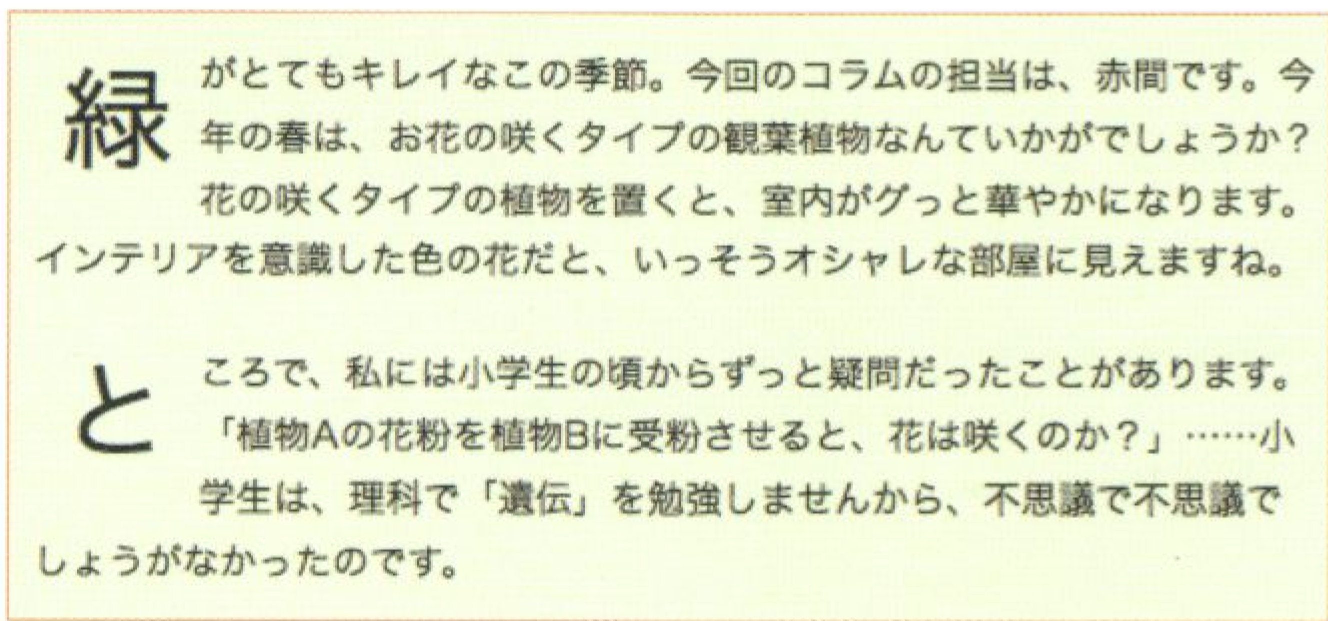
IE	P.E.	Safari	5.1 以上
Firefox	4 以上	Chrome	12 以上

※IEではtext-shadow、border-radiusに非対応。

TECHNIC 1

テキストにインパクトを出す

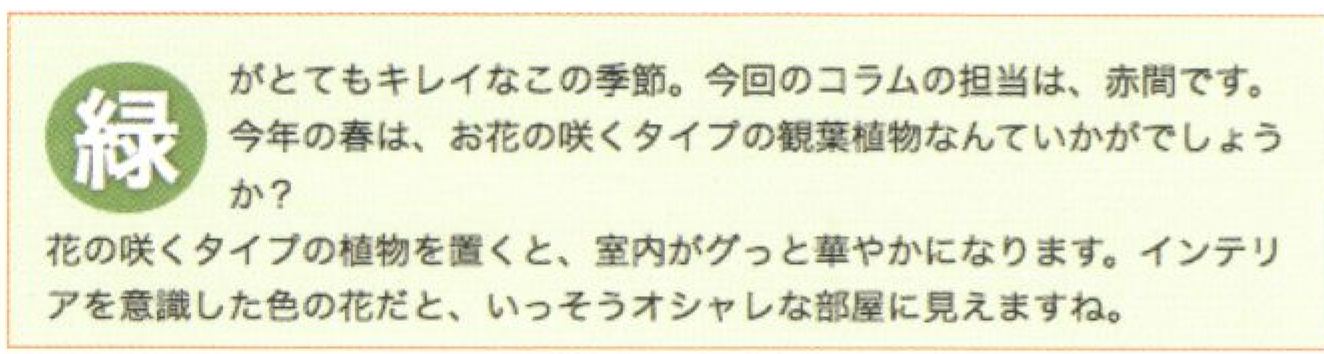
DropCapsを用いることで、段落の1文字目を目立たせることができます。

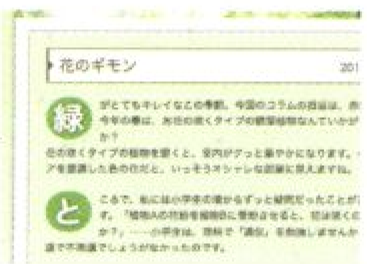


TECHNIC 2

CSS3を利用してより多彩な表現に

CSS3を使うことで、テキストシャドウなど、より多彩な表現を行うことができます。





ベースになる段落をシンプルにマークアップ

01

テキスト主体の段落がいくつかあるコラムページを例にしています。
特に1文字目に対するマークアップは意識せず、p要素でシンプルに段落としてマークアップします 01-1 01-2。

01-2 ブラウザで表示させたところ

緑がとてもキレイなこの季節。今回のコラムの担当は、赤間です。今年の春は、お花の咲くタイプの観葉植物なんていかがでしょうか？
花の咲くタイプの植物を置くと、室内がグッと華やかになります。インテリアを意識した色の花だと、いっそうオシャレな部屋に見えますね。

ところで、私には小学生の頃からずっと疑問だったことがあります。「植物Aの花粉を植物Bに受粉させると、花は咲くのか？」……小学生は、理科で「遺伝」を勉強しませんから、不思議で不思議でしょうがなかったのです。

01-1 シンプルなマークアップ

```
<div id="column">
<p>緑がとてもキレイなこの季節。今回のコラムの担当は、赤間です。今年の春は、お花の咲くタイプの観葉植物なんていかがでしょうか？<br />
花の咲くタイプの植物を置くと、室内がグッと華やかになります。インテリアを意識した色の花だと、いっそうオシャレな部屋に見えますね。</p>
<p>ところで、私には小学生の頃からずっと疑問だったことがあります。「植物Aの花粉を植物Bに受粉させると、花は咲くのか？」……小学生は、理科で「遺伝」を勉強しませんから、不思議で不思議でしょうがなかったのです。</p>
— 省略 —
<!--/column--></div>
```

02

基本となるCSSを準備します。
拡大する文字と、本文の文字サイズ・行間のバランスを調整するため、あらかじめコラム向けな、適切な値を設定します。
今回は基本文字サイズを14px、行間を1.6としました 02-1。

02-1 基本となるCSS

```
@charset "UTF-8";

body,h1,h2,h3,p,ul,li {
  margin: 0;
  padding: 0;
}
body {
  font-size: 14px;
  line-height: 1.6;
}
```

段落の1文字目にDropCapsを適用する

03

first-letter擬似要素で1文字目を選択します。「#column p:first-letter」とすることで、指定ID内のp要素すべてに適用することができます 03-1。

03-1 first-letter擬似要素で1文字目を選択

```
#column p:first-letter { }
```

TIPS

特定の段落だけに適用したい場合はnth-child擬似クラスを併用します。たとえば、「p:nth-child(odd):first-letter」と記述すると、奇数のp要素に対してDropCapsを適用することができます。
記述はnth-child、first-letterの順番にしないと適用されませんので注意が必要です。また、nth-childはIE9からの対応です。

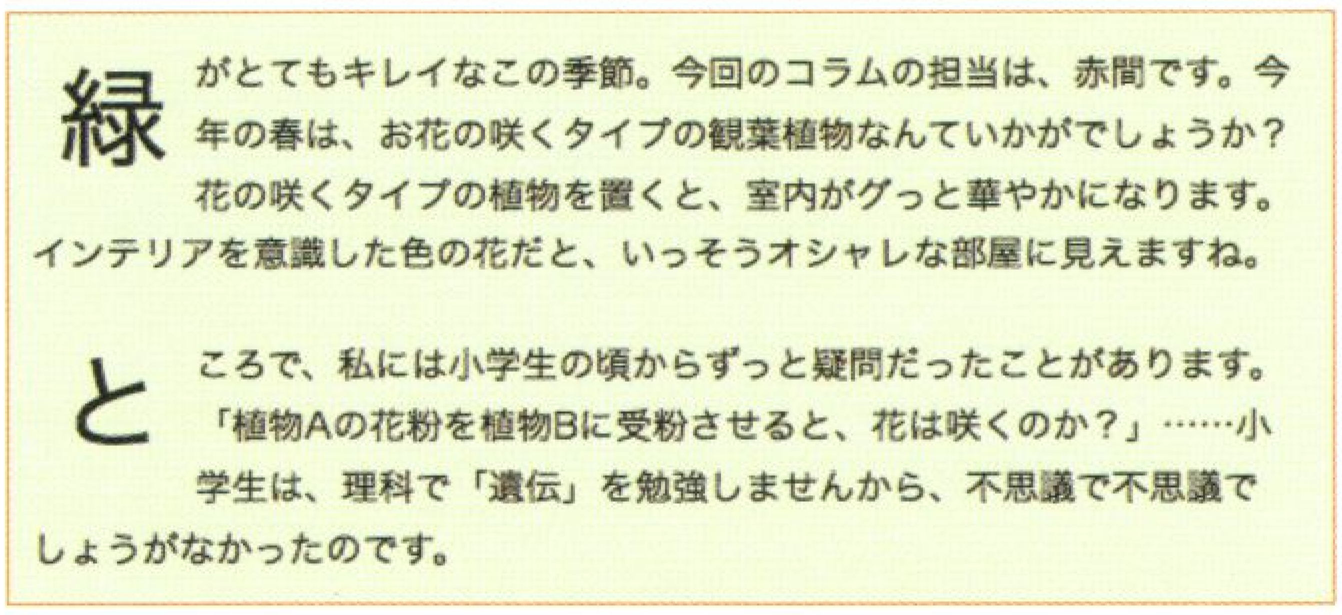
04

first-letter擬似要素で選択した１文字目のレイアウトやサイズ調整を行います 04-1 04-2。
まず、font-size: 3em;を指定し、１文字目を通常の段落の文字サイズより３倍の大きさに拡大します。
次にfloat: left;を指定し、１文字目を段落の左に回り込ませます。
さらにmargin-right: 10px;とpadding: 10px;を指定し、文字周りのバランスを整えます。
文字サイズの単位にはemを使うと“○文字分の大きさ”といった具合に、文字サイズのバランスをイメージしやすいでしょう。

04-1 １文字目のレイアウトやサイズ調整

```
#column p:first-letter {
  font-size: 3em;
  float: left;
  padding: 10px;
  margin-right: 10px;
}
```

04-2 DropCapsを適用した結果



05

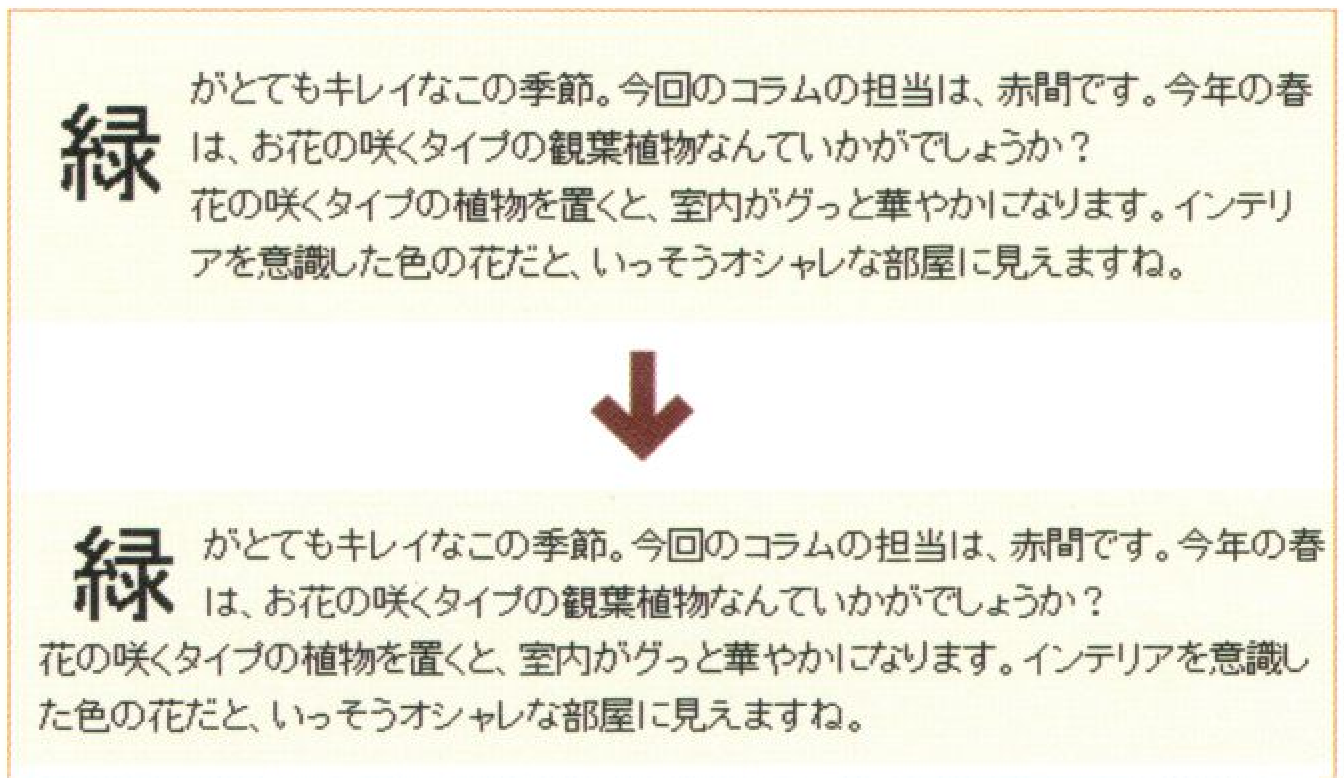
文字の回り込みバランスを適切にするため、line-height: 1;を指定します（IE6・7向けにはハックを使い *line-height: 1.5;） 05-1。
line-heightの指定がないと、IEやChromeなどのブラウザで１文字目の回り込みが適切に表現されない場合があります 05-2。必ずline-heightを指定するようにしましょう。

05-1 line-heightの指定

```
#column p:first-letter {
  font-size: 3em;
  float: left;
  padding: 10px;
  margin-right: 10px;
  line-height: 1;
  *line-height: 1.5; /*IE6,7用ハック*/
}
```

> **MEMO**
>
> line-heightの値を指定する際、単位は付けずに数値のみ記述しましょう。単位を付けない場合、「倍率」として扱われるので、たとえばline-height: 1;の場合は１倍となります。
> line-height: 1.5;と指定した場合、行間が1.5倍になるなど、数値と実際の行間のアキ具合をある程度想定した状態で指定することができます。

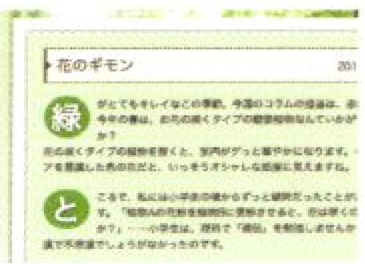
05-2 回り込みが適切になった



多彩な装飾を施す

06

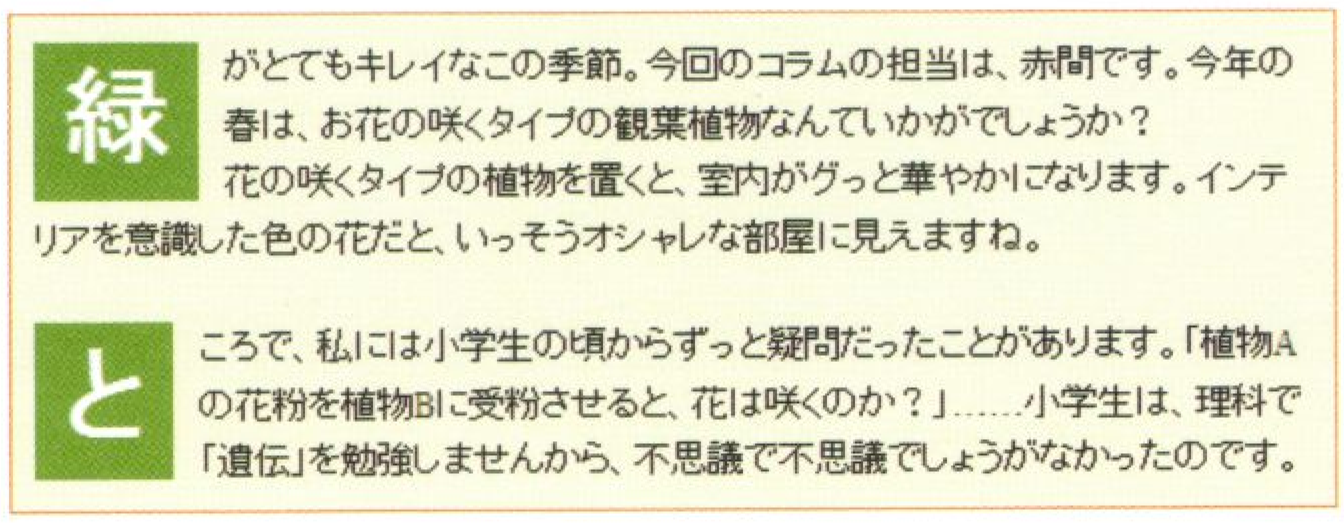
CSS3のプロパティを使って多彩な装飾を施してみます。
まずは、古いIE（バージョン６～８）においても、ある程度のデザインが表現できるようにします。
この例ではfont-weight: bold;、background: #49a342;、color: #fff;とし、文字の太さ、背景色、文字色を調整しました 06-1 06-2。



06-1 文字の太さ、背景色、文字色の調整

```
#column p:first-letter {
  font-size: 3em;
  float: left;
  padding: 10px;
  margin-right: 10px;
  line-height: 1;
  *line-height: 1.5; /*IE6,7用ハック*/
  font-weight: bold;
  background: #49a342;
  color: #fff;
}
```

06-2 文字の太さ、背景色、文字色を調整した結果



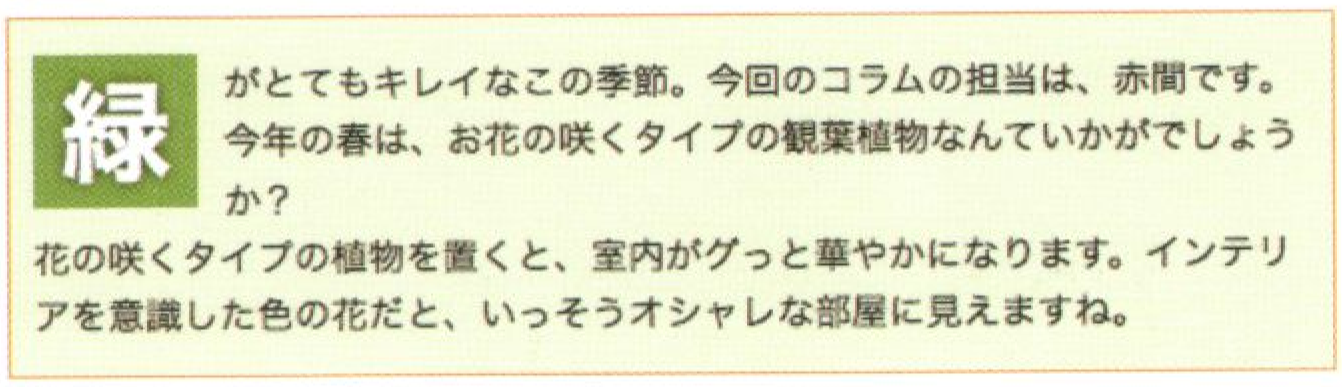
07

次に、CSS3による装飾です。
まずは文字に影を入れてみましょう。文字に影を指定するには、text-shadowプロパティを使用します。
今回は右下に落ちるような角度のある影ではなく、文字の周り全体に影が付くようにします。背景との区切りや、さりげなく目立たせるような指定として効果的です。
07-1のように指定します。x座標、y座標を0とすることで、文字の周りに均等な影を設定することができます07-2。

07-1 文字の周りに均等な影を設定

```
#column p:first-letter {
  font-size: 3em;
  float: left;
  padding: 10px;
  margin-right: 10px;
  line-height: 1;
  *line-height: 1.5; /*IE6,7用ハック*/
  font-weight: bold;
  background: #49a342;
  color: #fff;
  text-shadow: 0 0 5px #000;
}
```

07-2 均等な影を設定した結果



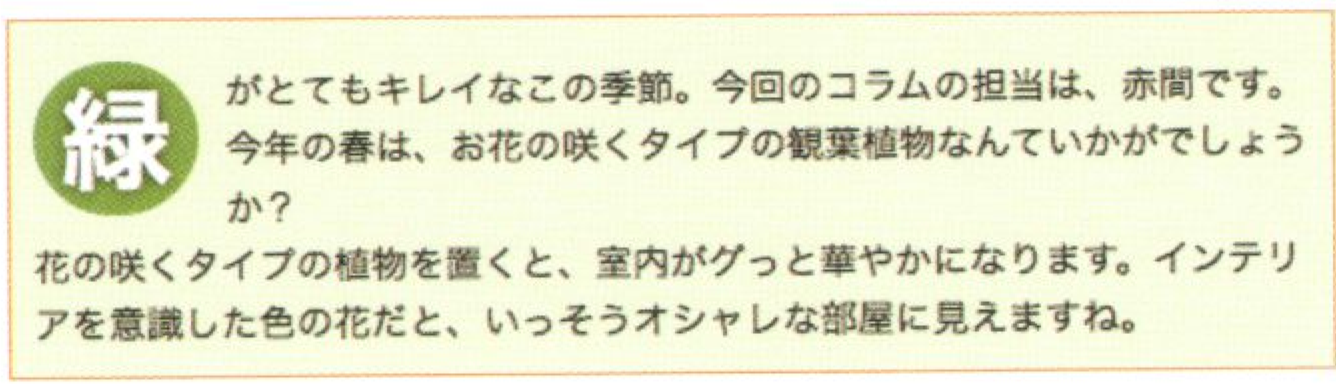
08

さらに背景も工夫してみましょう。
border-radiusを使用すると、ブロック要素に角丸の効果を与えることができます。
この例ではborder-radius: 30px;としました08-1。文字サイズに合わせて数値を大きくすることで、角丸長方形から円状に近づけて表現することができます08-2。

08-1 border-radiusで角丸の効果を与える

```
#column p:first-letter {
  font-size: 3em;
  float: left;
  padding: 10px;
  margin-right: 10px;
  line-height: 1;
  *line-height: 1.5; /*IE6,7用ハック*/
  font-weight: bold;
  background: #49a342;
  color: #fff;
  text-shadow: 0 0 5px #000;
  border-radius: 30px;
}
```

08-2 border-radiusの数値を大きくして円状に見せる



まとめ [CONCLUSION]

回り込みのレイアウト自体は古いバージョンのIEでも対応しており、簡素なテキストページにDropCapsの視覚効果を与えるだけでインパクトが出ます。HTML側のマークアップは極力シンプルなので、テキストや段落の追加時にも柔軟な対応が可能です。
レイアウト崩れ防止のため、IE向けにline-heightを必ず指定するようにしましょう。

09 ドロップシャドウ効果で写真にアクセントをつける



ドロップシャドウ（box-shadow）を使用すると、画像やボックス要素に対して影を付けることができます。ギャラリー系のサイトで写真をたくさん扱う場合など、アクセントとしてドロップシャドウが効果的です。

制作・文／赤間公太郎（株式会社マジカルリミックス）

制作のポイント

- box-shadowで写真にシャドウ効果を表現
- paddingで余白を調整し、写真の周りを装飾
- RGBAカラーモデルで影の濃淡を細かく調整

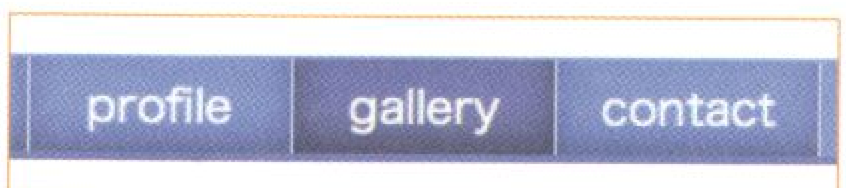
対応ブラウザ

IE	6 以上	Safari	5 以上
Firefox	4 以上	Chrome	12 以上

TECHNIC 1

ドロップシャドウで立体的に見せる

画像やボックス要素に影を付けることで、立体的な効果を出すことができます。





TECHNIC 2

ベベル的な表現も可能

影を内側に適用したり、座標の調整や影の色を工夫することで、CSS3だけでさまざまな見た目（ベベル的な表現など）を実現できます。





写真を並べたギャラリーを用意する

01

写真を並べたギャラリーサイトのHTMLを用意します。
ここでは 01-1 のように、li要素でマークアップします。
table要素を使うケースもありますが、table要素には“行”の概念があり、写真を追加・削除する際にHTMLソースの編集が煩雑になる場合があります。

01-1 li要素で写真を並べる

```
<ul>
<li><img src="img/photo/p01.jpg" alt="" /></li>
<li><img src="img/photo/p02.jpg" alt="" /></li>
<li><img src="img/photo/p03.jpg" alt="" /></li>
<li><img src="img/photo/p04.jpg" alt="" /></li>
<li><img src="img/photo/p05.jpg" alt="" /></li>
<li><img src="img/photo/p06.jpg" alt="" /></li>
<li><img src="img/photo/p07.jpg" alt="" /></li>
<li><img src="img/photo/p08.jpg" alt="" /></li>
<li><img src="img/photo/p09.jpg" alt="" /></li>
<li><img src="img/photo/p10.jpg" alt="" /></li>
<li><img src="img/photo/p11.jpg" alt="" /></li>
<li><img src="img/photo/p12.jpg" alt="" /></li>
</ul>
```

02

li要素にfloatを指定し、写真を横に並べます。写真のサムネイルサイズとメインカラムの幅を計算し、余白も含めて整えます。
ここでは3行4列のギャラリーですが、このようなレイアウトの場合、2行目で“強制改行”はせずに、1つの写真の横幅と全体のカラム幅を計算し、次の写真が“自然に2行目に落ちる”ようにします。そうすることで、HTMLに余計な記述が増えなくて済むうえ、並べ替えや写真追加にも柔軟に対応が可能です。
すべてのli要素に対して右marginを設定しましたが、行の最後（4枚目の写真が入るli要素）の右marginは、親要素のulに対してoverflow: hidden;を指定し、断ち切る設定にしています 02-1。
これにより、HTML側が極力少ないコードで、カラムの左右をバランスよく整えることができます 02-2。

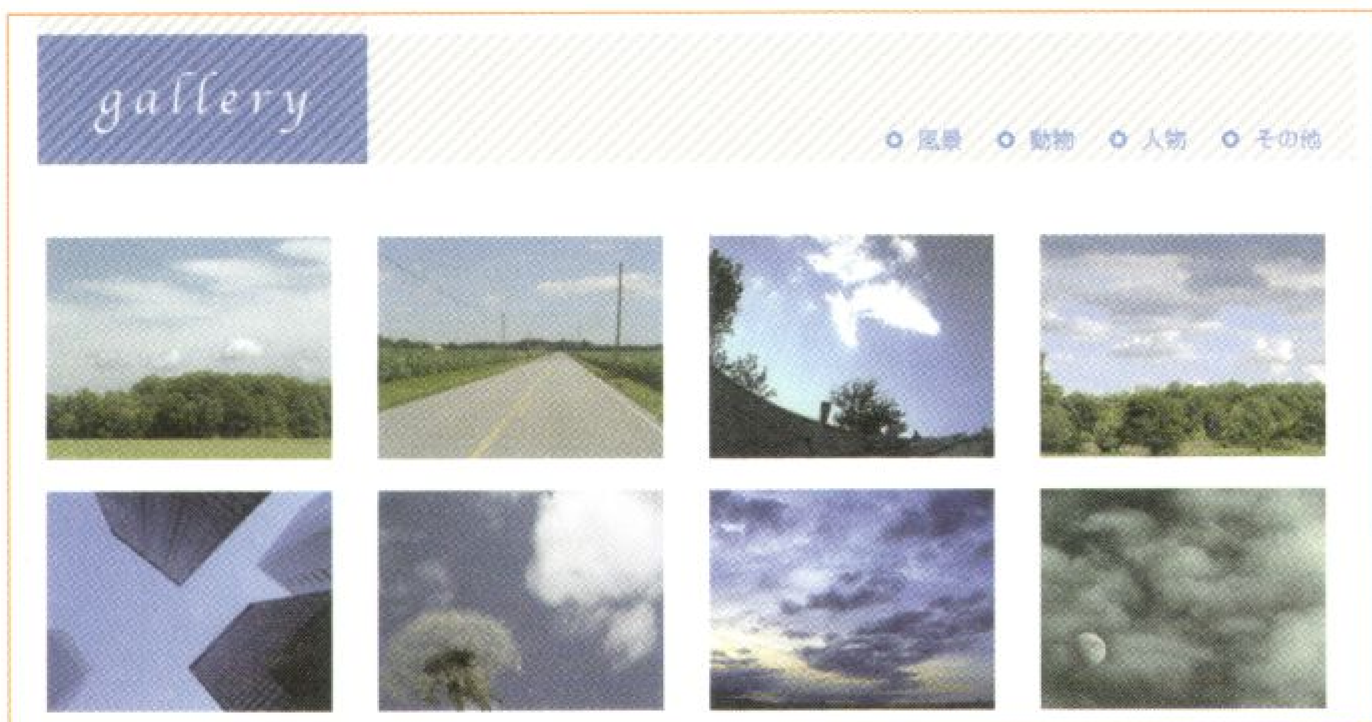
02-1 写真を横に並べ、余白も含めて整える

```
#content {
  overflow: hidden;
  margin: 40px 0;
}

— 省略 —

#content ul {
  width: 800px;
  margin: 5px 0 0 7px;
}
#content ul li {
  float: left;
  margin: 0 24px 20px 0;
}
```

02-2 カラムの左右がバランスよく整っている



MEMO

ページの横幅が800pxに対して、1枚あたりの写真の表示幅は187px（写真実寸175px・パディング10px・ボーダー2px）になり、写真を4枚並べると合計で748pxになります。物理的に5枚目は並びませんので、自然と2行目に“あえてカラム落ちさせる”設定としています。

03

写真の端からすぐにドロップシャドウをかけると、ゆとりのないデザインになるため、写真の周囲に余白や線で飾ります。写真に対してborder: 1px、padding: 5pxを設定し、写真の装飾を整えます。

paddingを加えたことにより、写真同士の間隔が広くなりました。02で設定したli要素に対して、右marginが24pxのままでは、3枚目の写真の後でカラム落ちしてしまいます。調整のため、li要素の右marginを12pxに変更しました03-1 03-2。

03-1 写真に枠を付け、さらに写真同士の間隔を調整

```
#content ul li {
  float: left;
  margin: 0 12px 20px 0;
}
#content ul li img {
  background: #fff;
  border: 1px solid #999;
  padding: 5px;
}
```

03-2 写真の装飾が整った



ドロップシャドウを適用する

04

box-shadowプロパティを使用して、写真に影を付けていきます。

基本的な書き方として、04-1のように、4つの数値＋色指定で構成します。

この表示結果は04-2のようになります。

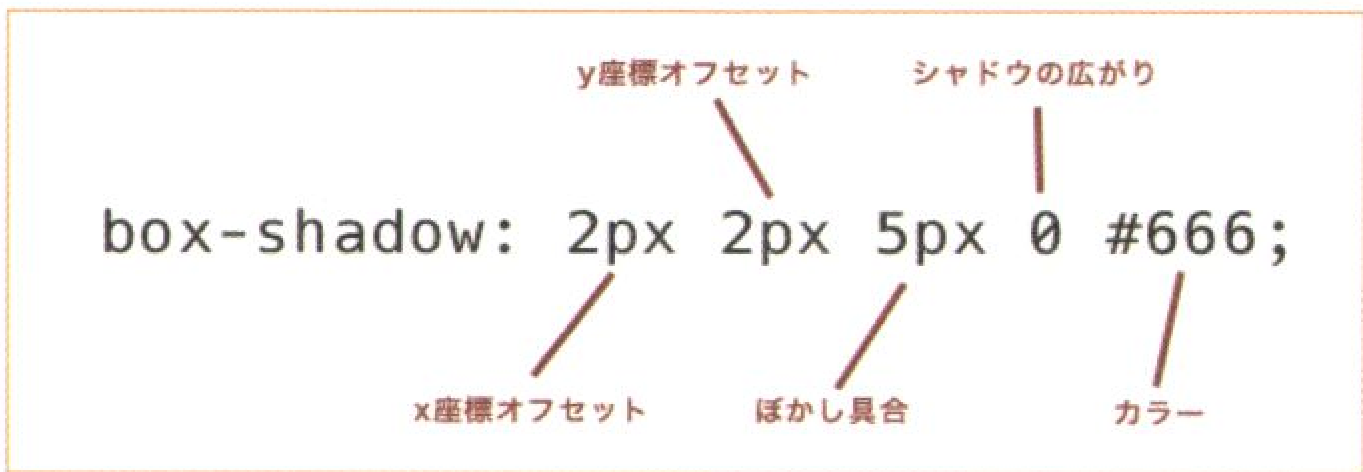
MEMO

box-shadowは数値部分を省略して記述することができます。

- 数値が4つの場合：x座標オフセット、y座標オフセット、ぼかし具合、シャドウの広がり
- 数値が3つの場合：x座標オフセット、y座標オフセット、ぼかし具合
- 数値が2つの場合：x座標オフセット、y座標オフセット

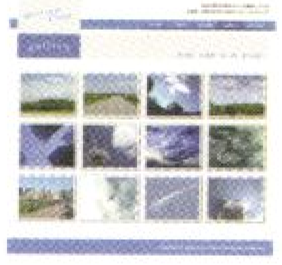
基本は4つの数値＋カラーで記述しますが、Web制作の現場でよく使われるのは数値が3つの書き方です。

04-1 box-shadowプロパティの基本的な書き方



04-2 写真の枠に影を付けたところ





*COLUMN

影を内側に適用する

box-shadowの値にinsetを追記すると、影を内側に適用することができます。
座標の調整や影の色を工夫すると、CSS3だけでさまざまな見た目を表現できます 01 02。

01 影のバリエーション



02 影のバリエーションを表現するコード

```
img.normal {
  background: #fff;
  padding: 15px;
  box-shadow: 2px 2px 5px #666;
}

img.inset {
  background: #fff;
  padding: 15px;
  box-shadow: 2px 2px 5px #666 inset;
}

img.bevel {
  background: #fff;
  padding: 15px;
  box-shadow: 0 -7px 10px 4px #ccc inset;
  -moz-border-radius: 10px;      /* Firefox */
  -webkit-border-radius: 10px;   /* WebKit */
  -o-border-radius: 10px;        /* Opera */
  -ms-border-radius: 10px;       /* IE10 */
  border-radius: 10px;           /* W3C */
}
```

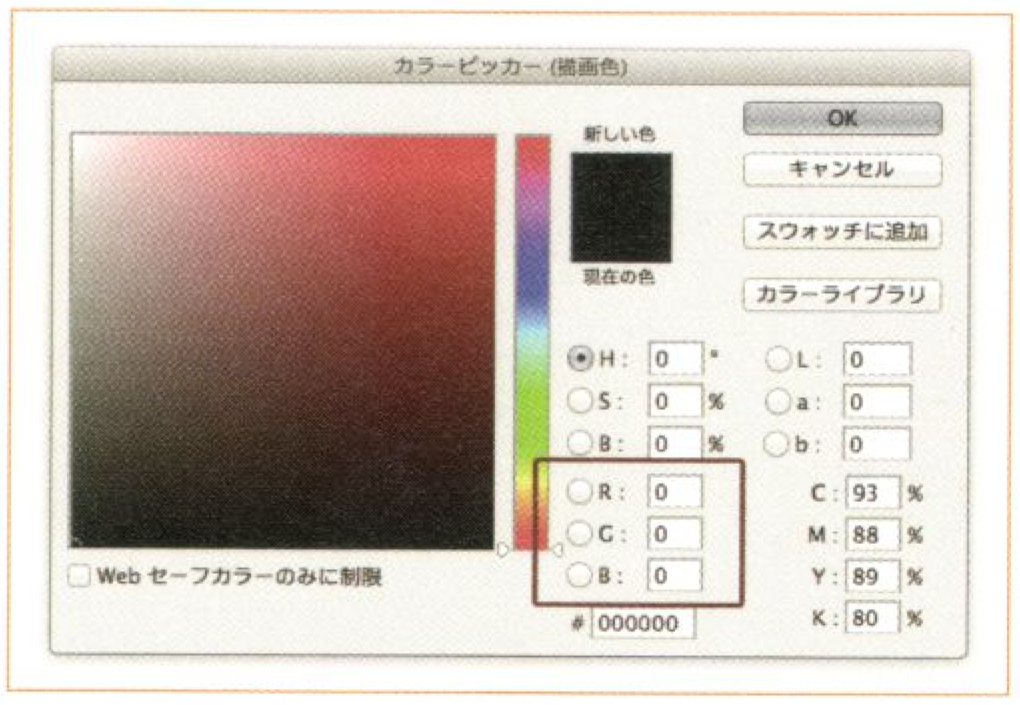
05

応用で、ナビゲーションにもドロップシャドウを適用してみましょう。
通常のソリッドなデザインのナビゲーションに対して、わずかに立体的な効果を適用します。
box-shadowプロパティの値にinsetを追加して、内側に影がつくようにします 05-1。
立体的に見せるポイントとしては、y座標にマイナスの値を指定すると、下の方の影の量が多くなります 05-2。

MEMO

シャドウの色指定は通常HEXカラー（16進数）で記述しますが、RGBAで指定することも可能です。
RBGAの場合、値をrgba(0, 0, 0, 0.5);のようにカンマ区切りで指定します。カンマ区切りの4つ目は、1=100%（不透明）ですので、影を半分の50%の濃さにしたい場合、0.5と記述します。
RGBの数値は、Photoshopなどのカラーピッカー 01 を見ると、設定がカンタンです。

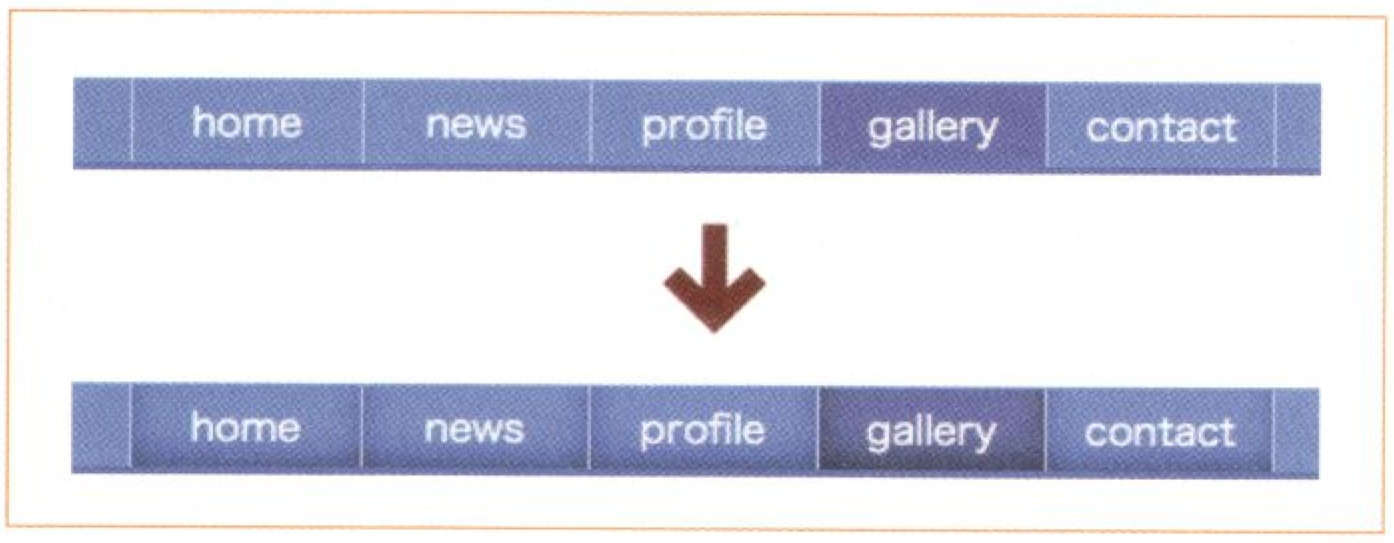
01 カラーピッカー



05-1 insetで内側に影がくるようにする

```
#header ul li a {
  display: block;
  width: 79px;
  height: 30px;
  line-height: 30px;
  box-shadow: 0 -5px 10px rgba(0, 0, 0, 0.5) inset;
}
```

05-2 下の方の影の量を多くすると立体的に見える



「CSS3 PIE」を使ってIE6 ～ 8に対応させる

06

box-shadowプロパティは本来IE9以降の対応ですが、JavaScriptライブラリを利用すると、IE6 ～ 8でも対応可能になります。
今回、ライブラリは「CSS3 PIE」を利用します。
06-1 06-2 のWebサイトからダウンロードし、圧縮ファイルを展開すると、いくつかのファイルがありますが、使用するファイルは「PIE.htc」です。

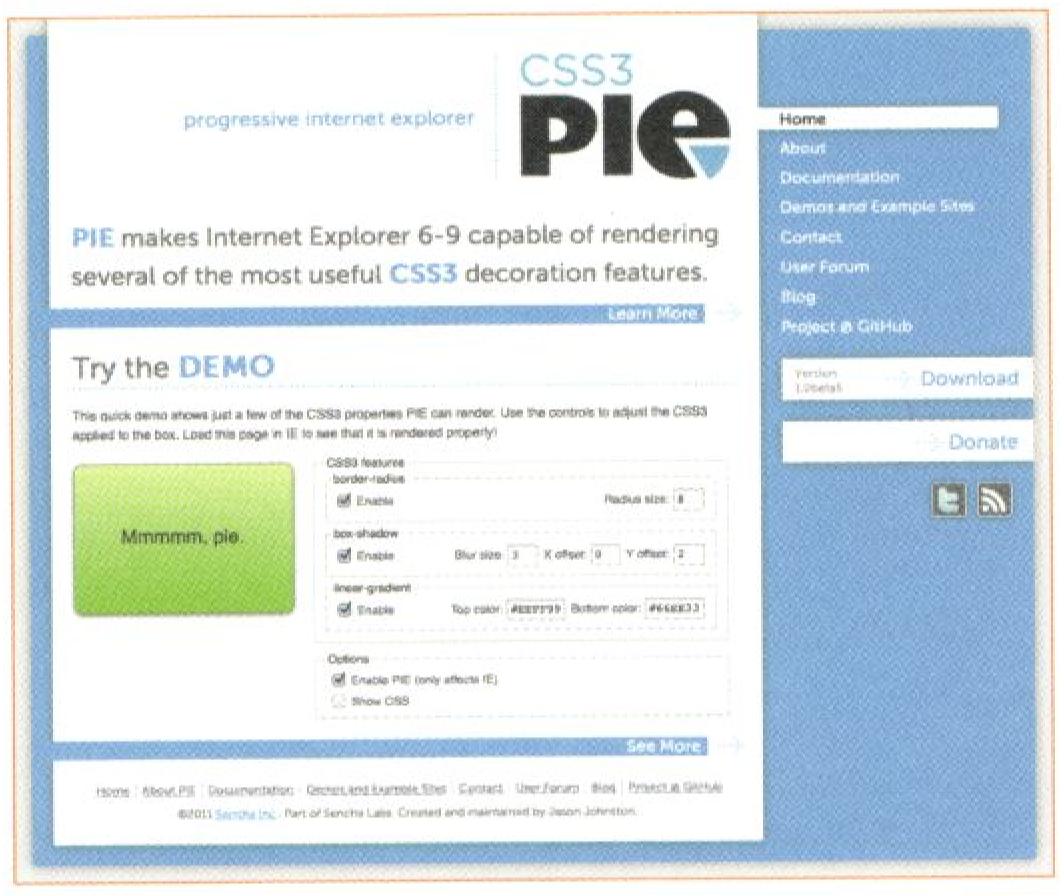
06-1 CSS3 PIE

http://css3pie.com/

MEMO

「PIE.htc」については、CHAPTER01-04「ウィンドウサイズに連動するリキッドタイプのフリーレイアウト」の53ページ、MEMOも参照。

06-2 「CSS3 PIE」のWebサイト



07

「CSS3 PIE」を適用するには、box-shadowプロパティを使用したスタイルの並びにbehavior: url("/PIE.htc");を記述します 07-1。

ここで注意が必要なのは、「PIE.htc」ファイルまでのパスです。通常は絶対パス（/から始まるパス）を指定しますが、本サンプルではローカル環境での確認もふまえ、相対パスを利用しています。

相対パスの利用時に、HTMLファイルが複数のディレクトリに分散するような構造の場合、それぞれのパスが変わってしまうので注意しましょう。
また、「CSS3 PIE」は"http://"から始まる指定もできません。

07-1 「CSS3 PIE」の適用

```
box-shadow: 2px 2px 5px #666;
behavior: url("PIE.htc");
```

ATTENTION!

「CSS3 PIE」はbox-shadowのinsetには対応していません。そのため、IE6 ～ 8からの対応は外側のドロップシャドウを設定する際に利用しましょう。

まとめ [CONCLUSION]

box-shadowの数値を細かく調整したり、複数のシャドウを組み合わせることにより、多彩な表現が可能です。特にナビゲーションやボタンなど、基本のHTMLやCSSにあまり手を入れることなく、視覚効果を追加できます。
古いバージョンのIEに対してはライブラリ利用で対応が可能ですが、ライブラリ独特の記述方法や、ほかのプロパティと衝突するケースも想定されますので、導入の際には注意しましょう。

最後に、ブラウザによるbox-shadowサポートの現状に触れておきます。
Firefox4以降、Safari5.1以降はベンダープレフィックスは不要です（ベンダープレフィックスについては、15ページのINTRODUCTION「ブラウザとベンダープレフィックス」の項もご覧ください）。
ただし、Safari5.0、Firefox3.6まではベンダープレフィックスが必要ですので、古いブラウザまでサポートする場合など、必要に応じて追記しましょう。

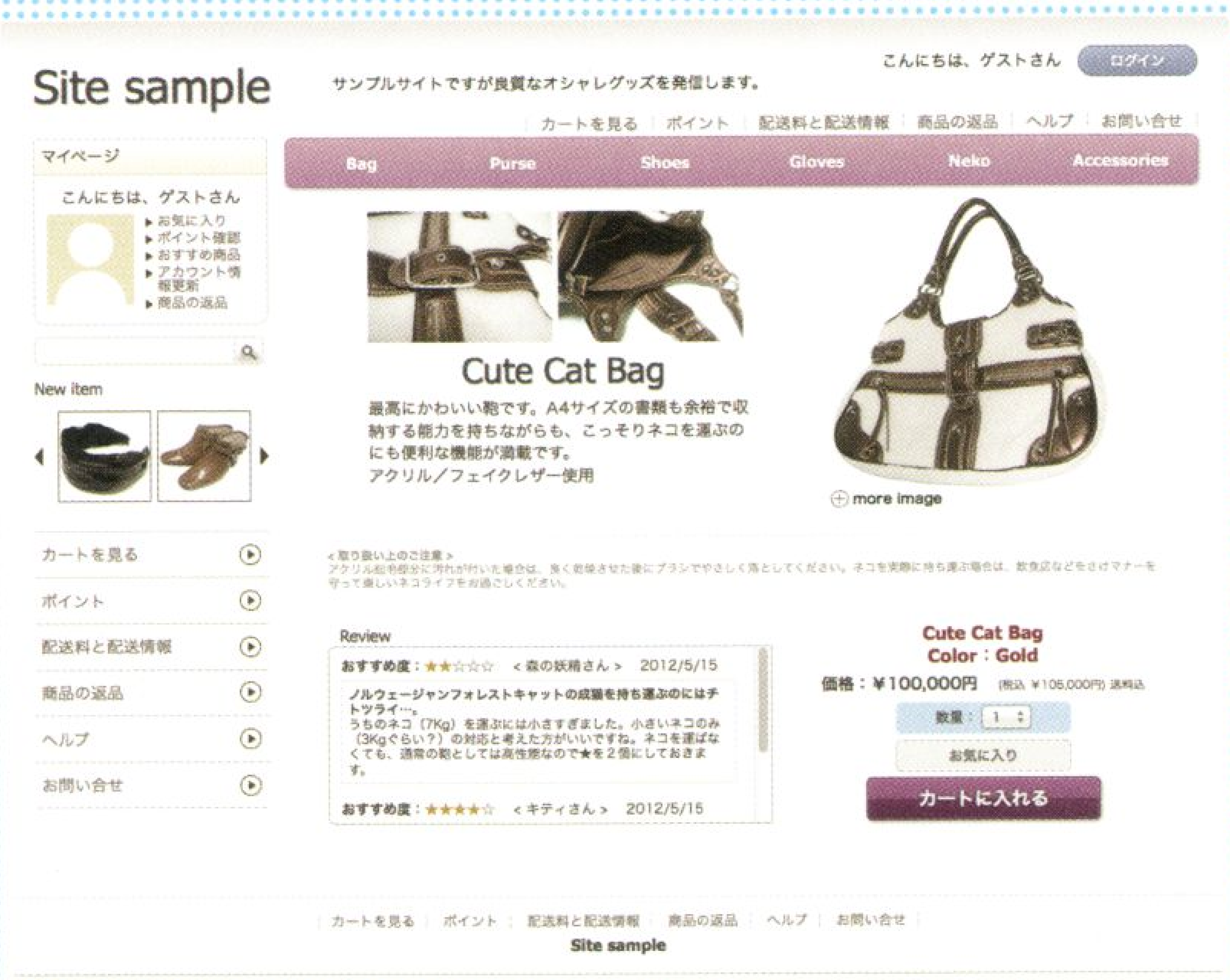
CHAPTER 04

デザインパーツ

Designparts

CSS3で表現するデザイン性豊かなボタン
HSLAによる色指定を使ったアイコンのカラーアニメーション
擬似クラスを利用した可読性の高いテーブル
マークアップのみで表現する階層のわかりやすいツリー図
borderと背景の組み合わせやoutlineで実現する飾り罫線
グラデーションと角丸で作る見出しのバリエーション
属性セレクタを活用したスマートフォンサイト
操作性を考慮したスマートフォン向けのパーツ
グラデーション機能を使った蛍光マーカー風の文字装飾

01 CSS3で表現するデザイン性豊かなボタン



CSS3のプロパティを生かして、つやのあるボタンやフラットなボタンを作ります。また、CSS3の対応度が低いIEについては、あえて外部ライブラリを使用せずに対応します。

制作・文／佐藤とも子・サチコ（螺旋デザイン）

制作のポイント

- つやのあるグラデーションの表現
- text-shadowとbox-shadowを使って影のニュアンスを表現
- IEでのアクセシビリティの確保

対応ブラウザ

IE	P.E.	Safari	5 以上
Firefox	8 以上	Chrome	16 以上

TECHNIC 1

画像を使わずにボタンを表現

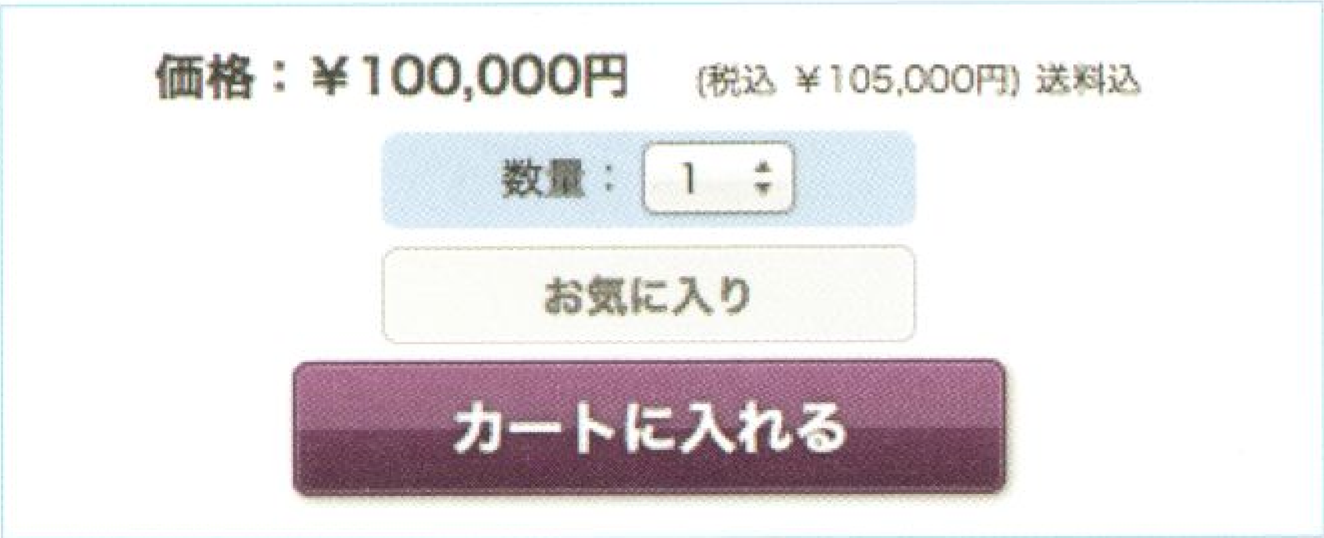
Webサイトでよく見かけるボタンを、画像を使わずに、box-shadowやグラデーションなどCSSだけで表現します。



TECHNIC 2

IE用表示に代替方法を使って表現

動作が重くなる外部ライブラリは利用せずにIEでの表示を保つ方法も検討しています。IE9ではグラデーションを表現することができないため、背景色を設定しています。





フラットなボタンを作る

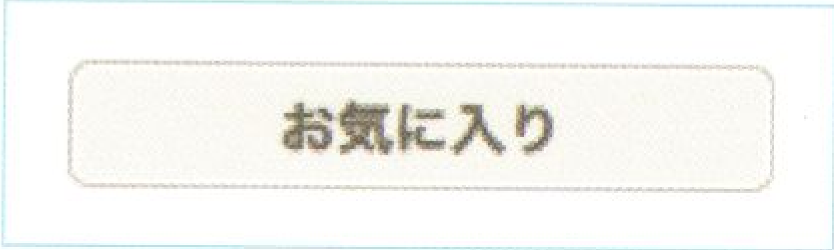
01

まずは、ベタ塗り・外側にラインが2本あるボタンを作ります 01-1。

HTMLは単純なaタグのみの状態です 01-2。

ここでは外側のddにidを付けていますが、この方がレイアウト調整などに便利だからです。

01-1 ボタンの原型



01-2 ボタンのHTML

```
<dd id="goodsFavorite"><a href="#">お気に入り</a></dd>
```

02

ボタン本体の親のddに、幅・marginを設定します。

幅の設定をここでしておくと、ボタン本体のa要素でpaddingやborderを付けてもddの幅より大きくならないので、CSSのコントロールが楽になります 02-1。

aタグをボタンにするには、はじめにdisplayをblockにし、次に背景とボーダーを設定しましょう 02-2。

文字設定などもしておきましょう。

角丸などの設定もした状態です 02-3。

02-1 幅・marginの設定

```
#mainContent #goodsSpec dd#goodsFavorite {
width:150px;
margin: 5px auto;
}
```

02-2 背景とボーダーの設定

```
#mainContent #goodsSpec dd#goodsFavorite a {
display:block;
background-color:#F0F0F0;
border: 1px solid #C1C1C1;
}
```

02-3 ボタンの角丸などを設定

```
#mainContent #goodsSpec dd#goodsFavorite a {
display:block;
background-color:#F0F0F0;
border: 1px solid #C1C1C1;
border-radius:5px;
padding: 3px 0;
text-align: center;
color: #616161;
text-decoration:none;
font-weight: bold;
font-size: 12px;
}
```

03

ボタン内側のラインを、box-shadowを使って表現します。

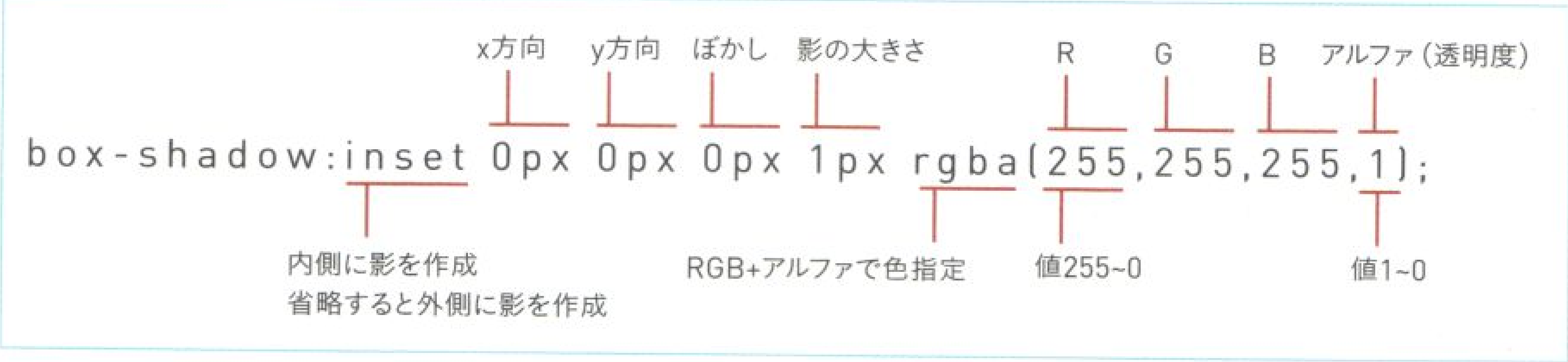
値の設定は 03-1 を参照してください。

これを利用して設定すると、03-2 のようになります。

最後に、マウスオーバー時の設定も作っておきましょう 03-3。

通常時a要素の設定を、:hoverのときに変わる部分を上書きしています。

03-1 box-shadowの値の設定



03-2 box-shadowの設定

```
#mainContent #goodsSpec dd#goodsFavorite a {
display:block;
background-color:#F0F0F0;
border: 1px solid #C1C1C1;
box-shadow:inset 0px 0px 0px 1px rgba(255,255,255,1);
border-radius:5px;
padding: 3px 0;
text-align: center;
color: #616161;
text-decoration:none;
font-weight: bold;
font-size: 12px;
}
```

03-3 マウスオーバー時の設定

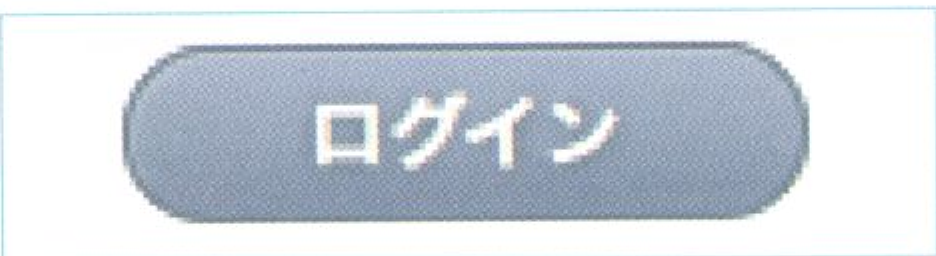
```
#mainContent #goodsSpec dd#goodsFavorite a:hover {
background-color:#B3A966;
box-shadow:inset 0px 0px 0px 1px rgba(220,217,193,1);
color: #FFF;
}
```

グラデーションのついたボタンの表現

04

グラデーションとハイライトのあるボタンを作ります04-1。
HTMLは、先ほどのフラットタイプのボタンと同じ構成です04-2。

04-1 グラデーションとハイライトのあるボタン



04-2 HTML

```
<dd><a href="#">ログイン</a></dd>
```

05

ボタンのベースは、先ほどのボタンと同じような感じで作ります05-1。
ハイライトも、先ほどのインラインの影と同じプロパティを利用します。
px指定の数値が、前のボタンは4つ（x、y、ぼかし、影のサイズ）でしたが、今回は3つになっています。この場合はx、y、ぼかしの指定になります05-2。
テキストのフチは、text-shadowで付けましょう05-3。text-shadowの値の設定も、box-shadowと同じような付け方で行えます。

05-1 ボタンのベース

```
#header dd a {
display:block;
width: 100px;
text-align:center;
font-weight:bold;
color: #FFF;
font-size: 12px;
line-height: 25px;
text-decoration: none;
border-radius: 13px;
border: 1px solid #5C6F91;
box-shadow:inset 1px 1px 0px
rgba(255,255,255,0.2);
}
```

05-2 box-shadowの設定

```
box-shadow:inset 1px 1px 0px rgba(255,255,255,0.2);
```

05-3 text-shadowの設定

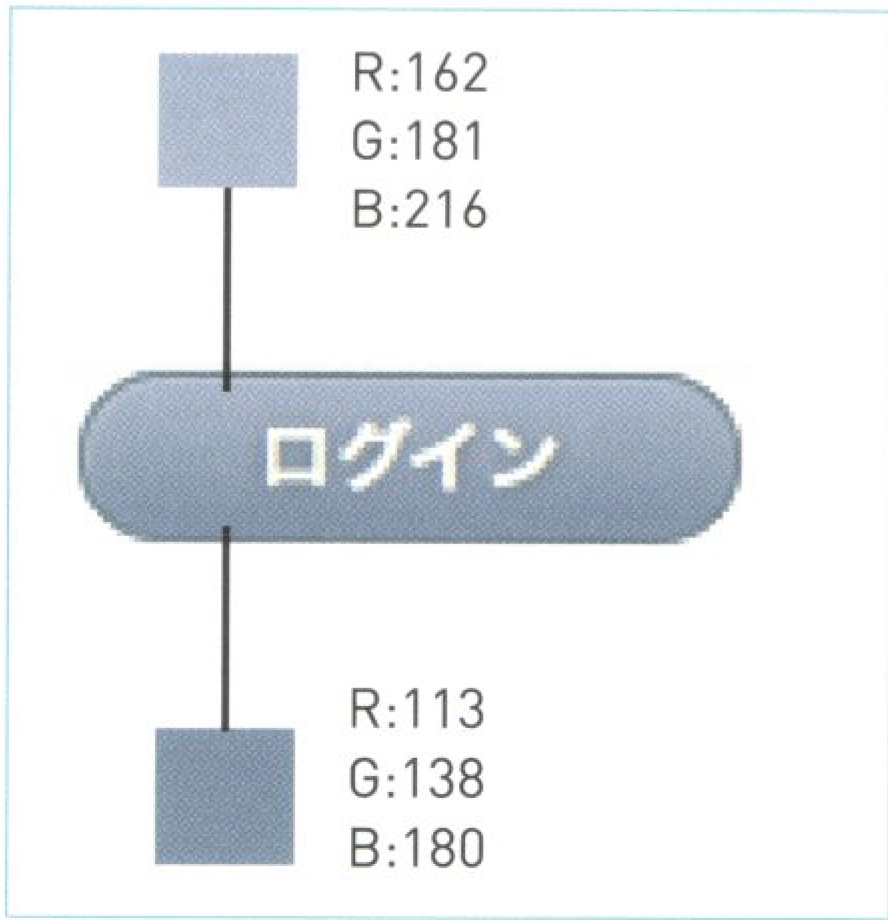
```
#header dd a {
display:block;
width: 100px;
— 省略 —
text-shadow:0px 0px 2px rgba(0,0,0,0.45);
— 省略 —
}
```



06

グラデーションを設定します。
ここでは06-1のように、2色のグラデーションになっています。
ベンダープレフィックスの設定が必要なので、接頭子の違う同じ記述を3回行います06-2。

06-1 グラデーションの設定



06-2 sample.css

```
/* Firefox */
background-image:
-moz-linear-gradient(
rgb(162,181,216),
rgb(113,138,180));

/* Webkit */
background-image:
-webkit-linear-gradient(
rgb(162,181,216),
rgb(113,138,180));

background-image:
linear-gradient(
rgb(162,181,216),
rgb(113,138,180));
```

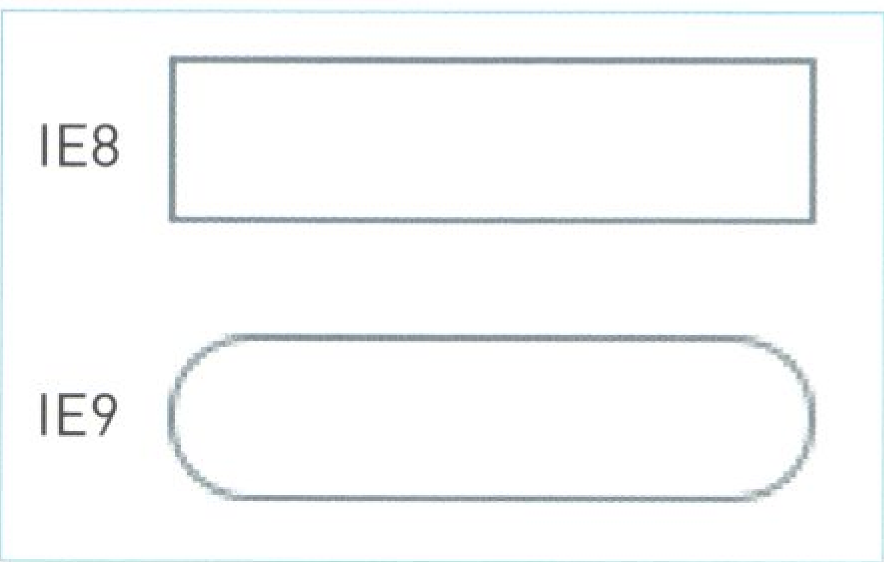
IE対策を考える

07

この状態で、IEで見てみるとどうでしょうか。IE9は角丸が生きていますが、IE8ではまったく設定が生きていません07-1。
IEでも角丸になるライブラリがありますが、動作が不安定だったり重かったりするので、ここではあえて使わないことにします。
IEには独自機能のfilterにグラデーション機能がありますので、それを使ってみます07-2。
フィルタを使ってみたところ、IE9では角丸を突き抜けてしまいました07-3。
外部ライブラリを利用せずに、できるだけIE対策をする場合は、以下のように対応することにします。

- IE9は角丸温存、グラデーションはあきらめる
- IE8は角丸をあきらめて、filterでグラデーションを再現する

07-1 IE8、IE9での表示



07-3 filterのグラデーション機能を適用



07-2 filterの設定（ie8.css）

```
filter:
  progid:DXImageTransform.Microsoft.gradient(startColorstr = #ffa2b5d8,endColorstr =
  #ff718ab4,GradientType = 0);

-ms-filter:"progid:DXImageTransform.Microsoft.gradient(startColorstr =
#ffa2b5d8,endColorstr = #ff718ab4,GradientType = 0)";
```

MEMO

CSS3の利用が向いているサイトのひとつとして、画像を極力使わずに素早い表示が求められるタイプのものがあります。すべてのブラウザでの表示を同じにするより、動画サイトなどに代表される、表示の快適さを優先するサイトです。
ここで紹介している、IE用のライブラリを使用しない方法も、快適さを優先するサイトの制作時などに参考にしてください。

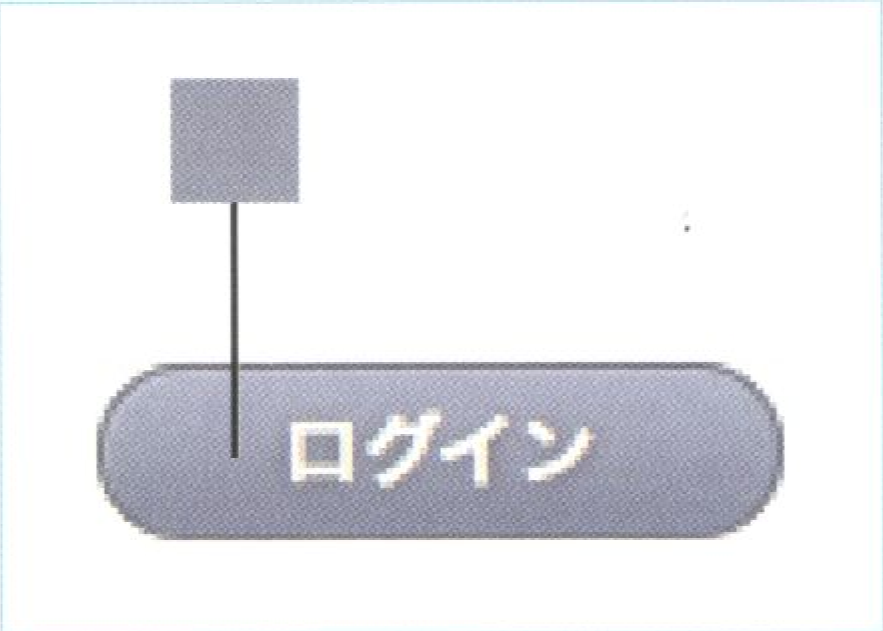
MEMO

上記のようにIEの過去バージョンなど、古いブラウザの見栄えを無理に揃えずに、CSS3などの最新技術に対応したモダンブラウザではよりリッチな見栄えを表現することをプログレッシブ・エンハンスメントと言います。
古いブラウザと新しいブラウザで、見た目が違っても同じ情報を得ることができ、無理にデザインを合わせないことによって軽快な動作を得ることができます。「プログレッシブ・エンハンスメント」については、INTRODUCTION「CSS3使用時の設計指針と注意点」の項目（16ページ）もご覧ください。

08

IE9対策の具体的な方法として、背景色を追記します。色はグラデーションの中から、メインの色に一番近い部分を選びましょう 08-1。
IE8以下のバージョン向けには、条件分岐で別CSSファイルを読み込ませて 08-2、そこにfilterの記述を入れます。
最終的には 08-3 のようになります。

08-1 背景色を選ぶ



08-2 IE8以下向けの案件分岐

```
<!--[if lte IE 8]>
<link rel="stylesheet" href="ie8.css" type="text/css" />
<![endif]-->
```

08-3 IE向け対策を施した状態



"つや"のあるグラデーションの表現

09

次は、つやのあるグラデーションのボタンを作ります 09-1。
HTMLはこれまでと同じ構成です 09-2。
先ほどのボタンのグラデーションに利用した色は2色ですが、今度は4色使います 09-3。

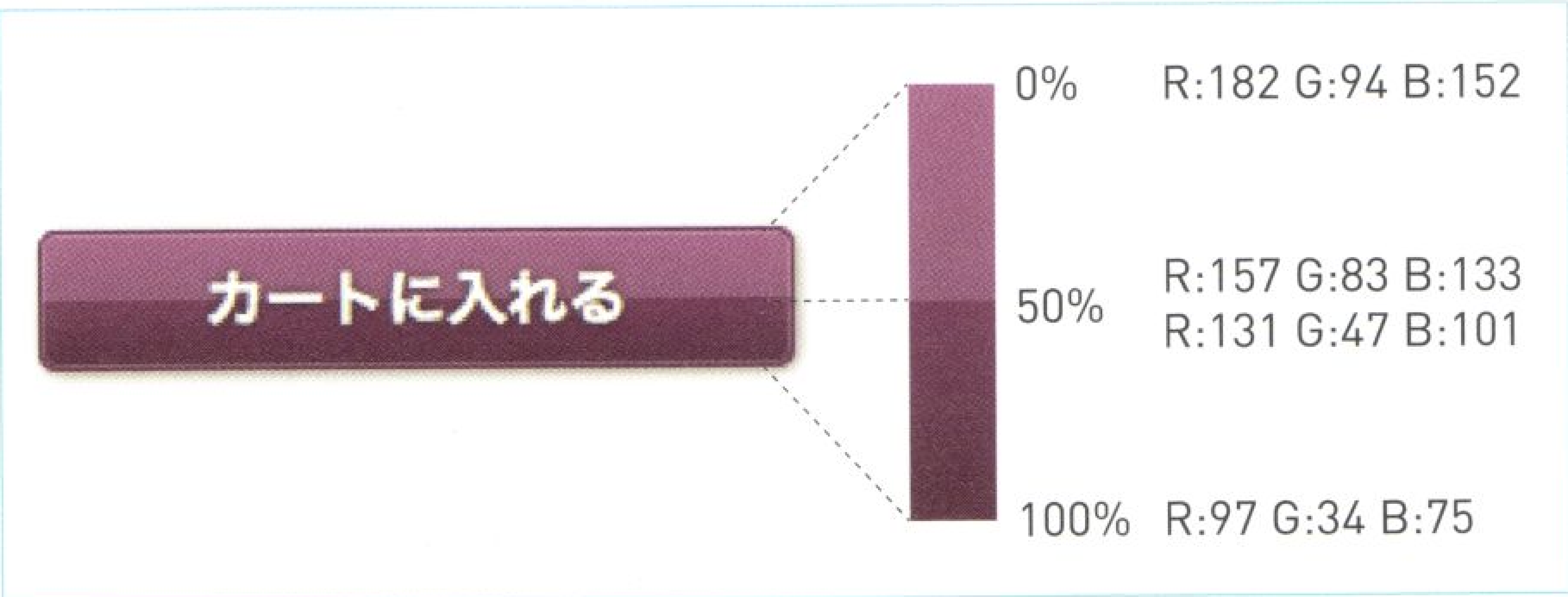
09-1 つやグラデーションのあるボタン



09-2 ボタンのHTML

```
<dd id="goodsCart"><a href="#">
カートに入れる</a></dd>
```

09-3 4色のグラデーション





10

3色以上のグラデーションを使う場合は、色設定の後ろに色の位置情報も入れます。今回は%を利用していますが、px指定もできます 10-1。
50%の値が2つありますが、こうすることによって、中央で色が切り替わったようなグラデーションを表現できます。

10-1 グラデーションの色・位置の設定

```
background-image:
linear-gradient(
rgb(182,94,152) 0%,
rgb(157,83,133) 50%,
rgb(131,47,101) 50%,
rgb(97,34,75) 100%);
```

TIPS

グラデーションの色設定には、rgb以外に、従来の#を利用した16進法での指定もできます。慣れ親しんだ16進法よりもrgbを使う利点のひとつは、アルファ値を含んだrgbaでの指定方法と親和性が高く、使い勝手がいいからです。

11

グラデーションの値指定に%を今回採用したのは、%でグラデーションの色位置が表示されるので、ボタンの高さが多少変わっても色の配分が変わらないからです 11-1。
同じ色合いで小さいボタンや大きいボタンを作成する場合も、グラデーションの値を変えずに完成できます。

11-1 ボタンの高さ変更で色の配分は変わらない



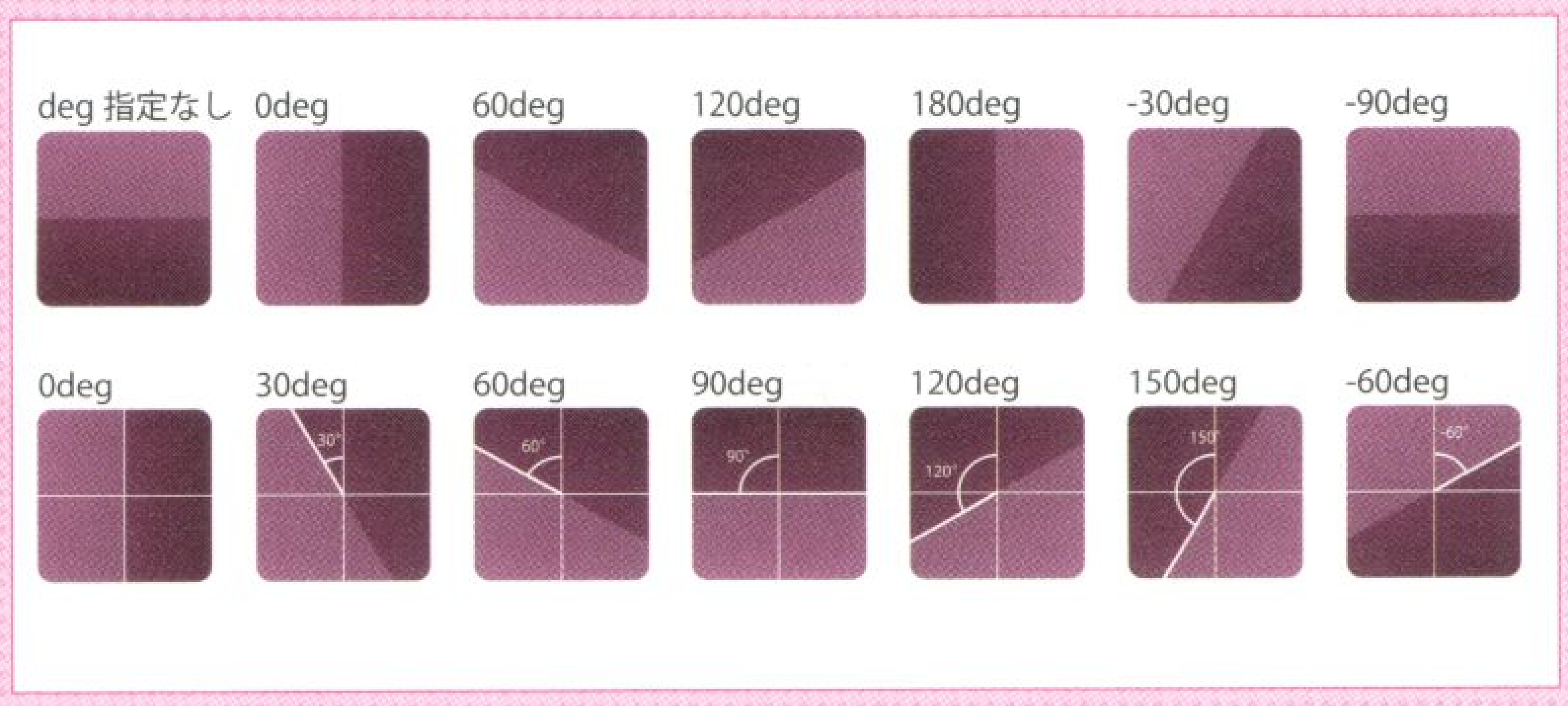
TIPS

今回のデザインは縦方向のグラデーションばかりでしたが、その他の角度のグラデーションも使うことができます。「20deg」のように色指定の前に角度を指定します 01。degというのはあまりなじみがない単位ですが、角度の単位の「度」と同じ意味です。

01 グラデーションの角度の指定

```
background-image:
linear-gradient(20deg,
rgb(182,94,152),
rgb(97,34,75));
```

02 さまざまなdegの値を入れてみたグラデーション



まとめ [CONCLUSION]

ボタンの大事な要素であるグラデーションは、ここで紹介した以上に、もっとさまざまな設定ができます。どんどん色を重ねてグロッシーな輝きを出したり、円状のグラデーションなどをつけたりも可能です。ただし、あまりやり過ぎると、対応していないブラウザとの差がかなり激しくなってしまうので、様子をみながら試していくことをおすすめします。

02 HSLAによる色指定を使ったアイコンのカラーアニメーション



↓



色相（H）・彩度（S）・明度（L）・不透明度（A／アルファ）で色指定をするHSLA使って、色の遷移がイメージしやすいカラーアニメーションを作成します。

制作・文／山本圭助
（株式会社ラナデザインアソシエイツ）

制作のポイント

- アルファ値（A）の指定
- 色がゆっくり変わっていくカラーアニメーション
- アニメーションの開始時間をずらすことで、波打つような表現も可能

対応ブラウザ

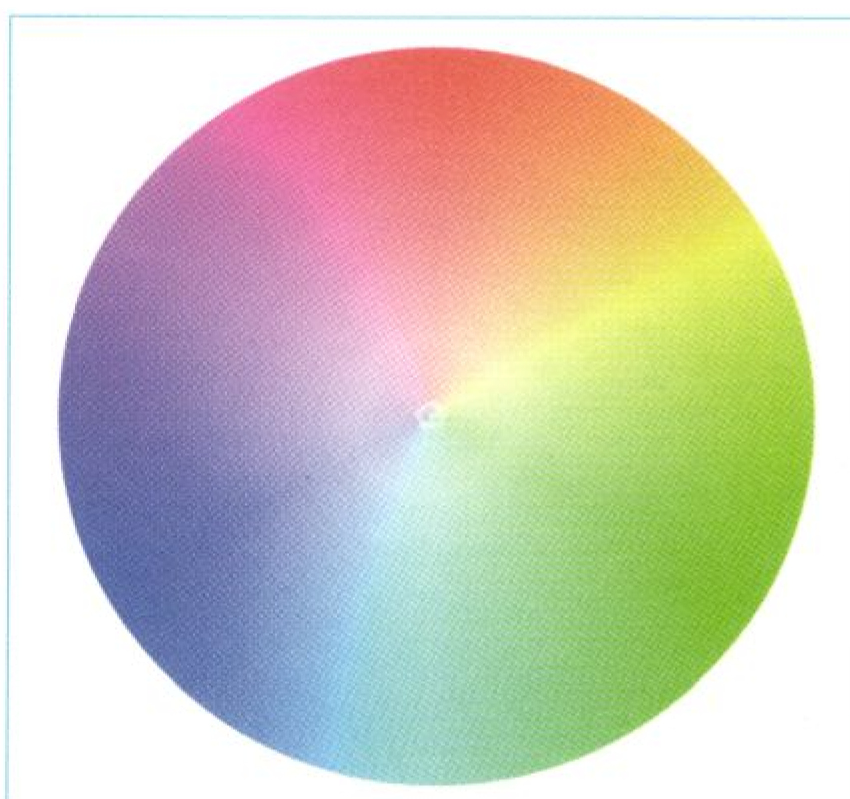
IE	P.E.	Safari	5 以上
Firefox	8 以上	Chrome	15 以上

※IEはanimationプロパティに非対応。

TECHNIC 1

HSLAによる色指定

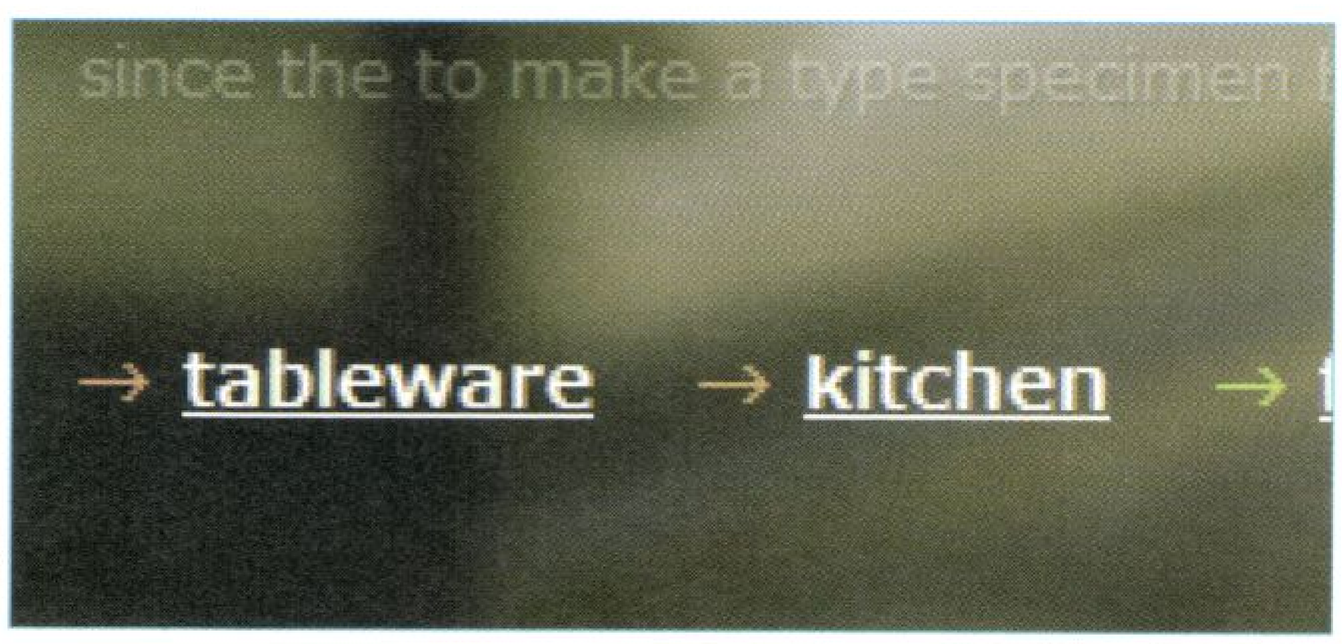
色相・彩度・明度・不透明度の4つの要素で色指定を行います。色の変化が直感的に理解できます。



TECHNIC 2

アニメーションの時間差再生

複数のオブジェクトに対して、色のグラデーションを時間差で再生することにより、波打つような表現が可能になります。



HSLAによるカラーの指定

01

CSSでは、01-1のように色を指定する方法がいろいろ用意されています。HSLAもそのうちのひとつで、色相と彩度と明度の色指定に透過度を加えたものです01-2。

色相（H）は、円形に色を並べたときの角度（0～359）で指定します。赤が0度、緑が120度、青が240度にあたります01-3。彩度（S）は0～100%で指定します。0%で灰色、100%で最も鮮やかな色になります。

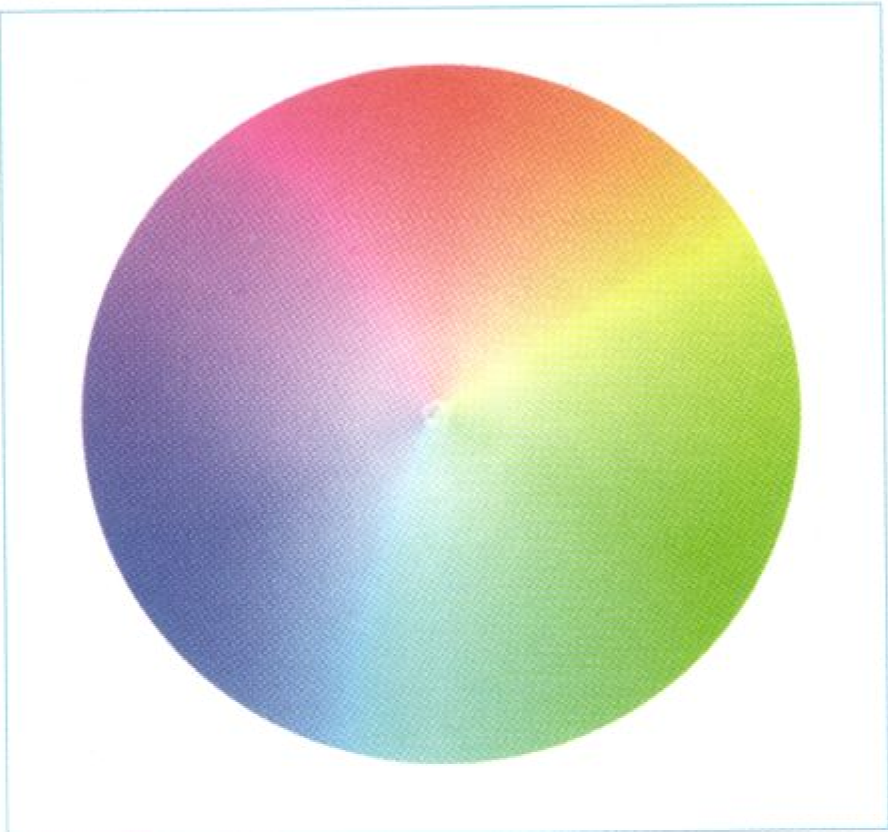
明度（L）は0～100%で指定します。0%で黒色、50%で色相で指定した色、100%で白色になります。

RGB方式よりも直感的に色をイメージでき、微妙な変更も加えやすいというメリットがあります。

01-1 CSSで色を指定する方法

```
color: red;
color: #f00;
color: #ff0000;
color: rgb(255, 0, 0);
color: rgba(255, 0, 0, 1);
color: hsl(0, 100%, 50%);
color: hsla(0, 100%, 50%, 1);
```

01-3 色相環



01-2 HSLAの意味と指定できる数値

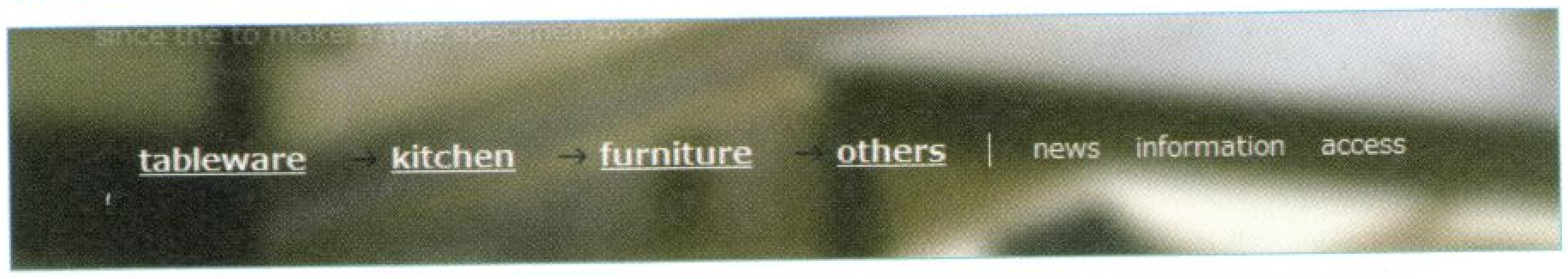
頭文字	要素	指定できる数値
H	Hue（色相）	0～359
S	Saturation（彩度）	0～100%
L	Lightness（明度）	0～100%
A	Alpha（不透明度）	0.0～1.0

02

リンクの文頭に付けたテキストアイコン（02-1の矢印部分）に、HSLAで色を指定します02-2 02-3。

IE8以下はHSL方式に非対応なので、その代替も記述しておきます。

02-1 リンク文頭のテキストアイコン

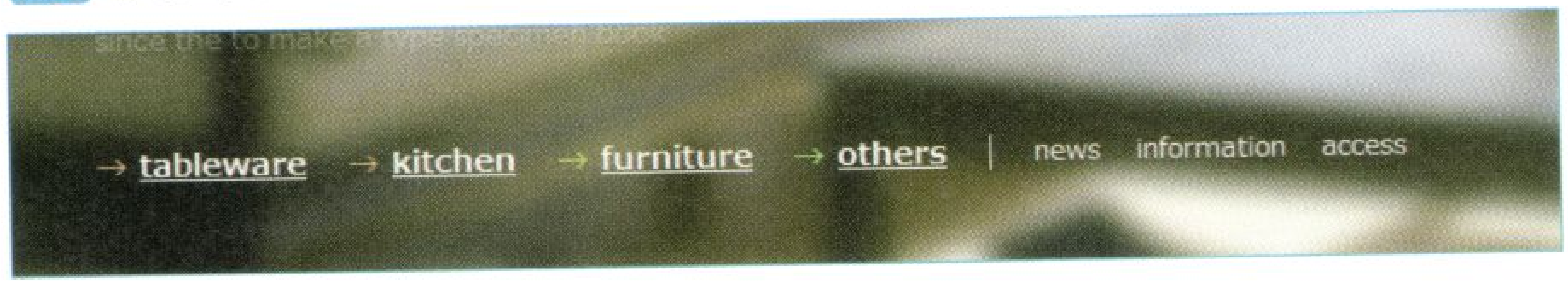


02-2 テキストアイコンにHSLAで色を付ける

```
// HTML
<ul>
  <li><span class="icon">→</span><a href="#">table ware</a></li>
  <li><span class="icon">→</span><a href="#">kitchen</a></li>
  <li><span class="icon">→</span><a href="#">furniure</a></li>
  <li><span class="icon">→</span><a href="#">others</a></li>
</ul>

// CSS
.icon {
  color: #f07575;
  color: hsla(0, 80%, 70%, 1);
}
```

02-3 テキストアイコンに色が付いた



03

次に、@keyframesで色相が順に変化していくアニメーション「color-anime」を作成します03-1。

03-1 色相が順に変化するアニメーション

```
// Chrome, Safari
@-webkit-keyframes color-anime {
  0% { color: hsla( 0, 80%, 70%, 1); }
  20% { color: hsla( 72, 80%, 70%, 1); }
  40% { color: hsla(144, 80%, 70%, 1); }
  60% { color: hsla(216, 80%, 70%, 1); }
  80% { color: hsla(288, 80%, 70%, 1); }
  100% { color: hsla(360, 80%, 70%, 1); }
}
```

```
// Firefox
@-moz-keyframes color-anime {
  0% { color: hsla( 0, 80%, 70%, 1); }
  20% { color: hsla( 72, 80%, 70%, 1); }
  40% { color: hsla(144, 80%, 70%, 1); }
  60% { color: hsla(216, 80%, 70%, 1); }
  80% { color: hsla(288, 80%, 70%, 1); }
  100% { color: hsla(360, 80%, 70%, 1); }
}
```

04

animationプロパティで、アニメーション「color-anime」を実行します04-1 04-2。

04-1 アニメーションを再生する

```
animation-name: color-anime;
animation-duration: 20s;
animation-timing-function: ease;
animation-delay: 0s;
animation-count: infinite;
```

04-2 ベンダープレフィックスを付ける

```
.icon {
  -webkit-animation: color-anime 20s ease 0s infinite;
  -moz-animation: color-anime 20s ease 0s infinite;
}
```

*COLUMN

CSSでアニメーション

@keyframesでアニメーションを作成し、animationプロパティで実行させます01 02。
一括で指定することもできます03。name以外は、省略するとデフォルト値が適用されます。

01 @keyframesの基本的な記述

```
@keyframes アニメーション名 {
  0% または from {
    開始。変更したいプロパティと値を記述。
  }
  100% または to {
    終了。変更したいプロパティと値を記述。
  }
}
```

02 animationプロパティ

要素	意味
animation-name:	@keyframesで指定したアニメーション名
animation-duration:	1回のアニメーションにかける時間。単位はs（秒）かms（ミリ秒）で、デフォルトは0s（0の時も単位が必要）
animation-timing-function:	アニメーション時のイージング。デフォルトはease
animation-delay:	開始までの待ち時間
animation-iteration-count:	実行回数。デフォルトは1。infiniteで無限回数も指定可能
animation-direction:	方向。デフォルトはnormalで一方方向。alternateで偶数回目のアニメーションを逆転
animation-fill-mode:	アニメーション実行前後に@keyframesの0%(from)や100%(to)に指定した値を適用するかどうか

03 animationプロパティを1行で指定

```
animation: name duration timing-function delay iteration-count direction fill-mode
```

05

4つあるリンクのアイコンを左から少しずつずらして実行していくことで、波打つようなアニメーションを表現します。
HTMLにクラス名を追加します 05-1。
次に、1つずつanimation-delay（アニメーションの開始時間の遅延）を設定します 05-2。
これで、アニメーションが2秒ずつ遅れて開始されるようになりました。

05-1 クラス名を追加する

```
<ul>
  <li class="list-0"><span class="icon">→</span><a href="#">table ware</a></li>
  <li class="list-1"><span class="icon">→</span><a href="#">kitchen</a></li>
  <li class="list-2"><span class="icon">→</span><a href="#">furniure</a></li>
  <li class="list-3"><span class="icon">→</span><a href="#">others</a></li>
</ul>
```

05-2 アニメーションを2秒ずつずらして実行する

```
.list-0 .icon { -webkit-animation-delay: 0s; -moz-animation-delay: 0s; }
.list-1 .icon { -webkit-animation-delay: 2s; -moz-animation-delay: 2s; }
.list-2 .icon { -webkit-animation-delay: 4s; -moz-animation-delay: 4s; }
.list-3 .icon { -webkit-animation-delay: 6s; -moz-animation-delay: 6s; }
```

*COLUMN

テキストアイコンをCSSで表示する

擬似クラス:beforeに対応したブラウザなら、02で紹介したコード 01 を 02 のように変更することで、CSSでテキストアイコンを表示することもできます。

01 変更前

```
// HTML
<ul>
  <li><span class="icon">→</span><a href="#">table ware</a></li>
  <li><span class="icon">→</span><a href="#">kitchen</a></li>
  <li><span class="icon">→</span><a href="#">furniure</a></li>
  <li><span class="icon">→</span><a href="#">others</a></li>
</ul>

// CSS
.icon {
  color: #f07575;
  color: hsla(0, 80%, 70%, 1);
}
```

02 変更後

```
// HTML
<ul>
  <li><a href="#">table ware</a></li>
  <li><a href="#">kitchen</a></li>
  <li><a href="#">furniure</a></li>
  <li><a href="#">others</a></li>
</ul>

// CSS
li:before {
  content: "→";
  color: #f07575;
  color: hsla(0, 80%, 70%, 1);
}
```

まとめ [CONCLUSION]

記述の簡易さから色指定は16進表記（#f00や#FF0000）や色言葉（redやwhite）を使うことが多いかと思いますが、HSL方式は直感的に色をイメージでき、微妙な変更も加えやすいので、マウスオーバー時などの変化やアニメーション時の指定などに色を決めやすくなります。
animationプロパティに対応しているブラウザはすべてHSLA指定にも対応しているので、積極的に使っていくとよいでしょう。

03 擬似クラスを利用した可読性の高いテーブル



ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ	-	-
2011/12/01	totoくじ購入代金	300	1回	300	-
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	-
2011/12/03	Edyチャージ	10,000	1回	10,000	-
2011/12/04	ソフマッピ	21,980	リボ	-	Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	-
2011/12/09	MDN COLLECTION	1,100	1回	1,100	-
2011/12/09	密林.CO.JP	5,527	1回	5,527	-
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ	-	-
2011/12/10	密林.CO.JP	3,872	1回	3,872	-
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	-
2011/12/13	スタイルアッププラザMDN	39,400	リボ	-	-
2011/12/13	ヨドバシデンキ	1,484	1回	1,484	-
2011/12/14	ウォーターサービス	3,570	1回	3,570	-
2011/12/15	MDN Design	6,615	1回	6,615	-
2011/12/15	MDN mTunes Store	450	1回	450	-
2011/12/15	MDNブックス	945	1回	945	-
2011/12/16	MDNカンパニーオンライン	11,487	1回	11,487	-
2011/12/17	セレクトショップMDN	5,505	1回	5,505	-
2011/12/17	ヘアーディメンションZ	12,900	1回	12,900	-
2011/12/22	MDNカーペット	3,543	1回	3,543	-
2011/12/23	南急ハンズ新宿店	5,240	1回	5,240	-
	合計	203,189	-	101,790	-

テーブルの可読性を上げるスタイリングは、これまでもプログラムでクラス名を付与すれば可能でしたが、擬似クラスを使うことで、HTMLに手を加えずにCSSのみでスタイリングができるようになります。これによって、開発工数の削減やメンテナンス性アップにつながります。

制作・文／山田敬美

制作のポイント

- 行は背景色、列はテキストの配置を指定する
- カーソルのある行をハイライトさせる
- 空のセルには別のスタイルを適用する

対応ブラウザ

ブラウザ	バージョン
IE	9 以上
Safari	3 以上
Firefox	3.5 以上
Chrome	2 以上

※上記は:nth-child()の対応ブラウザ。
※JavaScriptライブラリ使用時はIE6から対応。

TECHNIC 1

CSSのみで列に対してスタイルを指定

:nth-child(N)をtdに対して指定することで、列ごとに個別にスタイルを適用します。



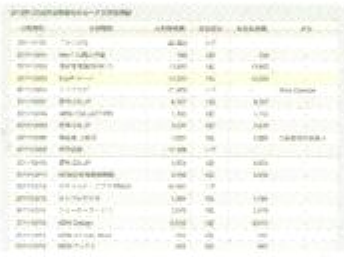
TECHNIC 2

空のセルにのみ指定した文字列を表示

擬似クラスの:emptyと、擬似要素の:beforeを組み合わせることで、空のセルにのみ指定した文字列を表示できます。



支払区分	お支払金額	メモ
リボ	-	-
1回	300	-
1回	15,800	-
1回	10,000	-
リボ	-	Time Capsule
1回	6,327	-
1回	1,100	-
1回	5,527	-
1回	3,620	ご挨拶用のお菓子
リボ	-	-



ベースとなるHTMLとCSSの記述

01

まずは01-1のような、ベースとなるシンプルなテーブルをHTMLとCSSで作成します01-2。

MEMO

以降のステップでは、このベースとなるHTMLを一切変更することなく、CSSのみで指定をしていきます。

01-1 ベースとなるテーブル



MdN Card こんにちは 山田敬美 さん（ログアウト） 前回ログイン日時：2011年12月25日 ヘルプ お問い合わせ

トップ ご利用明細 ご利用可能額 キャッシング お客様情報

カードご利用明細

確定済みの明細をご確認いただけます。

2012年1月25日お振替分のカードご利用明細

ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ		
2011/12/01	totoくじ購入代金	300	1回	300	
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	
2011/12/03	Edyチャージ	10,000	1回	10,000	
2011/12/04	ソフマッピ	21,980	リボ		Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	
2011/12/09	MDN COLLECTION	1,100	1回	1,100	
2011/12/09	密林.CO.JP	5,527	1回	5,527	
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ		
2011/12/10	密林.CO.JP	3,872	1回	3,872	
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	
2011/12/13	スタイルアッププラザMDN	39,400	リボ		
2011/12/13	ヨドバシデンキ	1,484	1回	1,484	
2011/12/14	ウォーターサービス	3,570	1回	3,570	
2011/12/15	MDN Design	6,615	1回	6,615	
2011/12/15	MDN mTunes Store	450	1回	450	
2011/12/15	MDNブックス	945	1回	945	
2011/12/16	MDNカンパニーオンライン	11,487	1回	11,487	
2011/12/17	セレクトショップMDN	5,506	1回	5,506	
2011/12/17	ヘアーディメンションZ	12,900	1回	12,900	
2011/12/22	MDNカーペット	3,543	1回	3,543	
2011/12/23	東急ハンズ新宿店	5,240	1回	5,240	
	合計	203,189		101,790	

プライバシーポリシー 利用規約 © MdN Card Co., Ltd.

01-2 ベースのHTML（必要な部分のみ抜粋）

```
<table>
<tr>
<td>2011/12/07</td><td>密林.CO.JP</td><td>6,327</td><td>1回</td><td>6,327</td><td></
td>
</tr>
<tr>
<td>2011/12/09</td><td>梅坂屋 上野店</td><td>3,620</td><td>1回</td><td>3,620</td><td>
ご挨拶用のお菓子</td>
</tr>
<tr>
<td>2011/12/09</td><td>有印良品</td><td>17,196</td><td>リボ</td><td></td><td></td>
</tr>
</table>
```

奇数行と偶数行で異なる背景色を指定

02

奇数行と偶数行で背景色を変えることで、それぞれの行を見やすくします02-1。これまでは、trタグにclass="odd"やclass="even"など、クラス名を個別に記述する必要がありましたが、CSS3の:nth-child(odd / even)を使えば、CSSだけで指定できます。
CSSの記述は02-2のとおり、とてもシンプルです。:nth-child(odd)で奇数行を、:nth-child(even)で偶数行を指定します。どちらか片方のみでも使えます。

MEMO

:nth-child(odd / even)では、「前からN番目」という指定のN部分に数字ではなく「odd」と「even」というキーワードを使うことで、奇数と偶数に対してスタイルを指定できます。

02-1 奇数行を白、偶数行をグレーにしたテーブル



MdN Card こんにちは 山田敬美 さん（ログアウト） 前回ログイン日時：2011年12月25日 ヘルプ お問い合わせ

トップ ご利用明細 ご利用可能額 キャッシング お客様情報

カードご利用明細

確定済みの明細をご確認いただけます。

2012年1月25日お振替分のカードご利用明細

ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ		
2011/12/01	totoくじ購入代金	300	1回	300	
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	
2011/12/03	Edyチャージ	10,000	1回	10,000	
2011/12/04	ソフマッピ	21,980	リボ		Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	
2011/12/09	MDN COLLECTION	1,100	1回	1,100	
2011/12/09	密林.CO.JP	5,527	1回	5,527	
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ		

02-2 nth-child(odd / even)で、奇数と偶数それぞれの背景色を指定

```
table tr:nth-child(odd) td {
    background: #fff;
}
table tr:nth-child(even) td {
    background: #f3f3f3;
}
```

TIPS

:nth-child(odd)と:nth-child(even)はそれぞれ、:nth-child(2n+1)、:nth-child(2n)と記述することもできます。

10行ごとにborderの太さを変更

03

10行ごとにborderを太くして目印にすることで、ひと目でだいたいの行数を把握できるようにします 03-1。連続する要素に対する繰り返しの指定をするには、:nth-child(An+B)を使います。

> **MEMO**
>
> :nth-child(An+B)では、B個目の要素に対する指定を、A個ごとに繰り返すことができます。

03-1 10行ごとにborderを太くしたテーブル


MdN Card こんにちは 山田敬美 さん（ログアウト）
前回ログイン日時：2011年12月25日
ヘルプ お問い合わせ
トップ ご利用明細 ご利用可能額 キャッシング お客様情報
カードご利用明細
確定済みの明細をご確認いただけます。
2012年1月25日お振替分のカードご利用明細

ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ		
2011/12/01	totoくじ購入代金	300	1回	300	
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	
2011/12/03	Edyチャージ	10,000	1回	10,000	
2011/12/04	ソフマッピ	21,980	リボ		Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	
2011/12/09	MDN COLLECTION	1,100	1回	1,100	
2011/12/09	密林.CO.JP	5,527	1回	5,527	
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ		
2011/12/10	密林.CO.JP	3,872	1回	3,872	
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	

04

CSSの指定方法は、04-1のとおりです。ここでは、10個目のtr要素に含まれるtdのborder-bottomを太くしています。それを10行ごとに繰り返すため、:nth-child(10n+10)という指定になります。

04-1 nth-child(An+B)で10行ごとにborder-bottomを3pxに指定

```
table tr:nth-child(10n+10) td {
    border-bottom: 3px solid #ccc;
}
```

列ごとにテキストの配置を指定

05

テーブルの列に対して、右寄せ左寄せなどのテキストの配置を指定したい場合、これまでは、td要素ひとつひとつに個別にクラス名を振らなければなりませんでした 05-1。しかし、:nth-child(N)を使えば、HTMLに手を加えることなく、列に対して自由にスタイルを指定できます 05-2。

05-1 nth-child(N)を使わない場合のHTML

```
<table>
<tr>
<td class="date">2011/12/07</td>
<td class="shop">密林.CO.JP</td>
<td class="price-1">6,327</td>
<td class="option">1回</td>
<td class="price-2">6,327</td>
<td class="memo"></td>
</tr>
<tr>
<td class="date">2011/12/09</td>
<td class="shop">梅坂屋 上野店</td>
<td class="price-1">3,620</td>
<td class="option">1回</td>
<td class="price-2">3,620</td>
<td class="memo">ご挨拶用のお菓子</td>
</tr>
<tr>
<td class="date">2011/12/09</td>
<td class="shop">有印良品</td>
<td class="price-1">17,196</td>
<td class="option">リボ</td>
<td class="price-2"></td>
<td class="memo"></td>
</tr>
</table>
```

05-2 列ごとにテキストの配置を指定したテーブル



MdN Card こんにちは 山田雅美 さん（ログアウト） 前回ログイン日時：2011年12月25日 ヘルプ お問い合わせ

トップ ご利用明細 ご利用可能額 キャッシング お客様情報

カードご利用明細

確定済みの明細をご確認いただけます。

2012年1月25日お振替分のカードご利用明細

ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ		
2011/12/01	totoくじ購入代金	300	1回	300	
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	
2011/12/03	Edyチャージ	10,000	1回	10,000	
2011/12/04	ソフマッピ	21,980	リボ		Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	
2011/12/09	MDN COLLECTION	1,100	1回	1,100	
2011/12/09	密林.CO.JP	5,527	1回	5,527	
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ		
2011/12/10	密林.CO.JP	3,872	1回	3,872	
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	
2011/12/13	スタイルアッププラザMDN	39,400	リボ		
2011/12/13	ヨドバシデンキ	1,484	1回	1,484	
2011/12/14	ウォーターサービス	3,570	1回	3,570	
2011/12/15	MDN Design	6,615	1回	6,615	
2011/12/15	MDN mTunes Store	450	1回	450	
2011/12/15	MDNブックス	945	1回	945	
2011/12/16	MDNカンパニーオンライン	11,487	1回	11,487	
2011/12/17	セレクトショップMDN	5,505	1回	5,505	
2011/12/17	ヘアーディメンションZ	12,900	1回	12,900	
2011/12/22	MDNカーペット	3,543	1回	3,543	
2011/12/23	南急ハンズ新宿店	5,240	1回	5,240	
	合計	203,189		101,790	

プライバシーポリシー 利用規約 © MdN Card Co., Ltd.

MEMO

:nth-child(N)では、()内で指定したN個目の要素に対してスタイルを指定できます。

***COLUMN**

colgroup要素やcol要素へのtext-align

列をグループ化するcolgroup要素やcol要素へのtext-alignの指定は、IE8から使えなくなりました。これは、W3Cの仕様に則った正しい実装で、ほかのモダンブラウザでも同様に機能しません。

06

CSSの指定方法は **06-1** のとおりです。左から順番に数えてN番目のtd要素に対して、text-alignプロパティを使い、テキストの配置を指定します。

06-1 nth-child(N)でテキストの配置を指定

```
table td:nth-child(1) { text-align: center; }
table td:nth-child(2) { text-align: left; }
table td:nth-child(3) { text-align: right; }
table td:nth-child(4) { text-align: center; }
table td:nth-child(5) { text-align: right; }
table td:nth-child(6) { text-align: left; }
```

ATTENTION!

セルを一部結合しているなどの複雑なテーブルの場合は、指定した順番がズレてしまうため、この方法は使えません。

カーソルのある行をハイライト

07

カーソルのある行をハイライトすることで、横軸のデータを見やすくします **07-1**。これまでは、JavaScriptでhover時にクラス名を追加するなどで対応していましたが **07-2**、ユーザーアクション擬似クラスの:hoverを使うことで、CSSのみで対応可能です。

MEMO

:hover擬似クラスはCSS3ではなく、CSS2から利用可能な擬似クラスです。IEでも、a要素のみであればIE6から、その他の要素に対してはIE7から実装されています。

07-1 カーソルのある行を黄色くハイライトしたテーブル



MdN Card こんにちは 山田雅美 さん（ログアウト） 前回ログイン日時：2011年12月25日 ヘルプ お問い合わせ

トップ ご利用明細 ご利用可能額 キャッシング お客様情報

カードご利用明細

確定済みの明細をご確認いただけます。

2012年1月25日お振替分のカードご利用明細

ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ		
2011/12/01	totoくじ購入代金	300	1回	300	
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	
2011/12/03	Edyチャージ	10,000	1回	10,000	
2011/12/04	ソフマッピ	21,980	リボ		Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	
2011/12/09	MDN COLLECTION	1,100	1回	1,100	
2011/12/09	密林.CO.JP	5,527	1回	5,527	
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ		
2011/12/10	密林.CO.JP	3,872	1回	3,872	
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	
2011/12/13	スタイルアッププラザMDN	39,400	リボ		
2011/12/13	ヨドバシデンキ	1,484	1回	1,484	
2011/12/14	ウォーターサービス	3,570	1回	3,570	
2011/12/15	MDN Design	6,615	1回	6,615	
2011/12/15	MDN mTunes Store	450	1回	450	
2011/12/15	MDNブックス	945	1回	945	
2011/12/16	MDNカンパニーオンライン	11,487	1回	11,487	
2011/12/17	セレクトショップMDN	5,505	1回	5,505	
2011/12/17	ヘアーディメンションZ	12,900	1回	12,900	
2011/12/22	MDNカーペット	3,543	1回	3,543	
2011/12/23	南急ハンズ新宿店	5,240	1回	5,240	
	合計	203,189		101,790	

プライバシーポリシー 利用規約 © MdN Card Co., Ltd.

07-2 :hoverを使わない場合のHTML

```
<table>
<tr>
<td>2011/12/07</td><td>密林.CO.JP</td><td>6,327</td><td>1回</td><td>6,327</td><td></td>
</tr>
<tr class="hover"><!--JavaScriptで.hoverを付与-->
<td>2011/12/09</td><td>梅坂屋 上野店</td><td>3,620</td><td>1回</td><td>3,620</td><td>ご挨拶用のお菓子</td>
</tr>
<tr>
<td>2011/12/09</td><td>有印良品</td><td>17,196</td><td>リボ</td><td></td><td></td>
</tr>
</table>
```

08

CSSの指定方法は08-1のとおりです。現在カーソルがあるtr要素（行）に含まれるtd要素の背景色を変更します。

08-1 :hoverで現在カーソルのある行をハイライト

```
table tr:hover td {
    background: #fdfbe0;
}
```

ATTENTION!

:hoverは他の記述よりも後に来るよう記述しないと、動かない場合があります。

空のセルに文字列を表示

09

:empty擬似クラスを使うと、中身がない空のセルに対してのみ、特定のスタイルを指定することができます。これを利用して、空のセルに「-」という文字列を表示させた上で、そのセルのみ中央揃えにします09-1。これまでは、プログラムで文字列を挿入するか、手作業で記述する必要がありましたが、今回はCSSのみで実装します。

09-1 空のセルに「-」を表示したテーブル



ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ	-	-
2011/12/01	totoくじ購入代金	300	1回	300	-
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	-
2011/12/03	Edyチャージ	10,000	1回	10,000	-
2011/12/04	ソフマッピ	21,980	リボ	-	Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	-
2011/12/09	MDN COLLECTION	1,100	1回	1,100	-
2011/12/09	密林.CO.JP	5,527	1回	5,527	-
2011/12/09	柳坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ	-	-
2011/12/10	密林.CO.JP	3,872	1回	3,872	-
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	-
2011/12/13	スタイルアッププラザMDN	39,400	リボ	-	-
2011/12/13	ヨドバシデンキ	1,484	1回	1,484	-
2011/12/14	ウォーターサービス	3,570	1回	3,570	-
2011/12/15	MDN Design	6,615	1回	6,615	-
2011/12/15	MDN mTunes Store	450	1回	450	-
2011/12/16	MDNブックス	945	1回	945	-
2011/12/16	MDNカンパニーオンライン	11,487	1回	11,487	-
2011/12/17	セレクトショップMDN	5,505	1回	5,505	-
2011/12/17	ヘアーディメンション2	12,900	1回	12,900	-
2011/12/22	MDNカーペット	3,543	1回	3,543	-
2011/12/23	南急ハンズ新宿店	5,240	1回	5,240	-
	合計	203,189	-	101,790	-

ATTENTION!

IEに擬似クラスを対応させるプラグイン（selectivizr.js。後述）を利用した場合でも、IE6とIE7では:beforeには対応していないため、空のままになります。しかし、表示が崩れたり、意味が通じなくなるわけではないため、特別に対応はしていません。

10

CSSの指定方法は10-1のとおりです。1つ目のtd:emptyで空のセルのみ中央揃えにし、2つ目のtd:empty:beforeで「-」という文字列を出力します。

*COLUMN

擬似要素とは

:beforeは擬似クラスと同じようにコロン（:）から始まりますが、「擬似要素」という種別です。擬似クラスが要素の状態に対してスタイルを適用できるのに対し、擬似要素は要素内の特定の文字に対するスタイル指定や、HTML上に存在しないコンテンツ（文字や画像など）を追加できるという特徴があります。

MEMO

!importantをつけることで、06でのテキスト配置の指定よりも:emptyでの指定を優先させています。

10-1 空のセルに「-」を表示し、中央揃えに変更

```
table td:empty {
    text-align: center !important;
}
table td:empty:before {
    content: "-";
}
```

Internet Explorerへの対応

11

CSS3の擬似クラスはIE9からの実装となり、それ以前のIEでは利用することができませんが、「selectivizr.js」というプラグインを使うことで、IE8以下でもCSS3の擬似クラスを使ったスタイル指定を適用できるようになります 11-1 。
(selectivizr.jsの説明については、CHAPTER01-04「ウィンドウサイズに連動するリキッドタイプのフリーレイアウト」の54ページ、MEMOも参照。)

11-1 selectivizr.js適用前（上）と適用後（下）のIE8の表示



2012年1月25日お振替分のカードご利用明細

ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ		
2011/12/01	totoくじ購入代金	300	1回	300	
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	
2011/12/03	Edyチャージ	10,000	1回	10,000	
2011/12/04	ソフマッピ	21,980	リボ		Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	
2011/12/09	MDN COLLECTION	1,100	1回	1,100	
2011/12/09	密林.CO.JP	5,527	1回	5,527	
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ		
2011/12/10	密林.CO.JP	3,872	1回	3,872	
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	



2012年1月25日お振替分のカードご利用明細

ご利用日	ご利用先	ご利用金額	支払区分	お支払金額	メモ
2011/11/21	ファンクル	22,823	リボ	-	-
2011/12/01	totoくじ購入代金	300	1回	300	-
2011/12/02	激安家電販売なにわ	15,800	1回	15,800	-
2011/12/03	Edyチャージ	10,000	1回	10,000	-
2011/12/04	ソフマッピ	21,980	リボ	-	Time Capsule
2011/12/07	密林.CO.JP	6,327	1回	6,327	-
2011/12/09	MDN COLLECTION	1,100	1回	1,100	-
2011/12/09	密林.CO.JP	5,527	1回	5,527	-
2011/12/09	梅坂屋 上野店	3,620	1回	3,620	ご挨拶用のお菓子
2011/12/09	有印良品	17,196	リボ	-	-
2011/12/10	密林.CO.JP	3,872	1回	3,872	-
2011/12/12	MDN生命保険保険料	3,505	1回	3,505	-

> **MEMO**
> ここで取り上げた擬似クラスに関してはIE9で実装されていますが、IE9がすべての擬似クラスに対応しているというわけではありません。

12

使い方は簡単です。12-1 のURLからselectivizr.jsをダウンロードし、HTMLのhead内で読み込むだけで設置できます 12-2 。

> **ATTENTION!**
> selectivizr.jsを読み込む前に、jQueryなどのJavaScriptライブラリを読み込んでおく必要があります。

12-1 selectivizr.js

http://selectivizr.com/

12-2 head内でjQueryとselectivizr.jsを読み込む

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
<!--[if (gte IE 6)&(lte IE 8)]><script src="selectivizr.js"></script><![endif]-->
```

> **MEMO**
> IE以外のブラウザやIE9以上では必要がないため、条件分岐コメントを使ってIE6からIE8に対してのみ読み込まれるよう指定します。

まとめ [CONCLUSION]

今回はテーブルに絞って解説をしましたが、擬似クラスはそれ以外にもさまざまな使い方ができる便利な機能です。CSS3を実務で使うときにネックとなる古いIEでも、JavaScriptライブラリを使えばほぼ同じようにスタイルを適用できるため、今から使えるCSS3のひとつといえるでしょう。擬似クラスを使って、これまで実現できなかったようなUIにもチャレンジしてみましょう。

04 マークアップのみで表現する階層のわかりやすいツリー図



階層構造がひと目でわかりやすいツリー図をマークアップのみで作成する方法を紹介します。リストの装飾にはCSS3を使って、IEでも表示が保持されるようJavaScriptを利用しました。

制作・文／中西祐介
（株式会社ワン・トゥー・テン・デザイン）

制作のポイント

- シンプルなマークアップでリストを表現
- JavaScriptでIEの表示にも対応
- linear-gradientなどのCSS3を使って装飾

対応ブラウザ

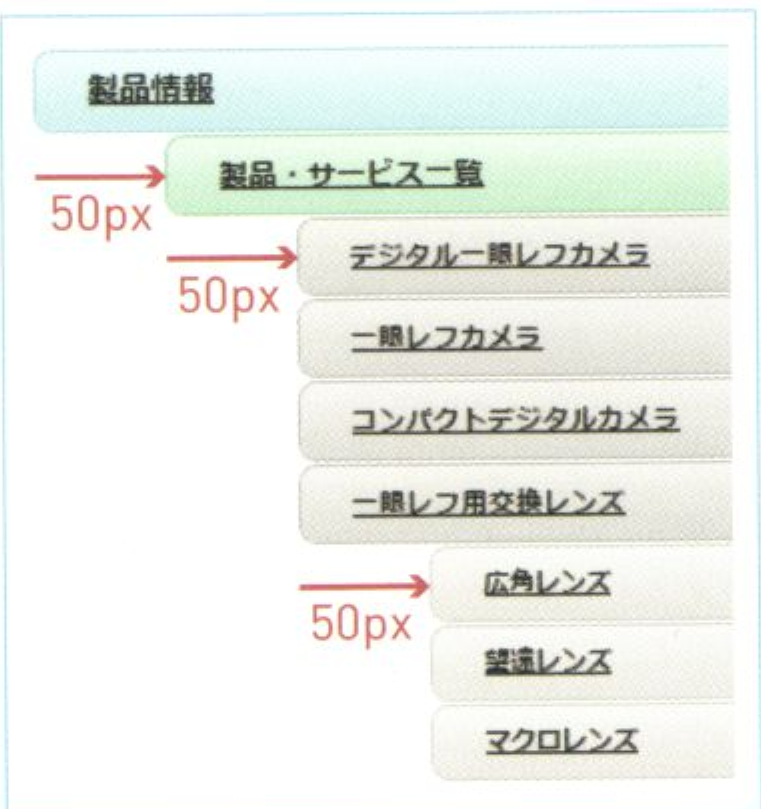
ブラウザ	バージョン
IE	6 以上
Safari	5 以上
Firefox	4 以上
Chrome	10 以上

※IEではborder-radius（IE6～8）、linear-gradient（IE6～9）に非対応。

TECHNIC 1

余白を空けることで階層を表現

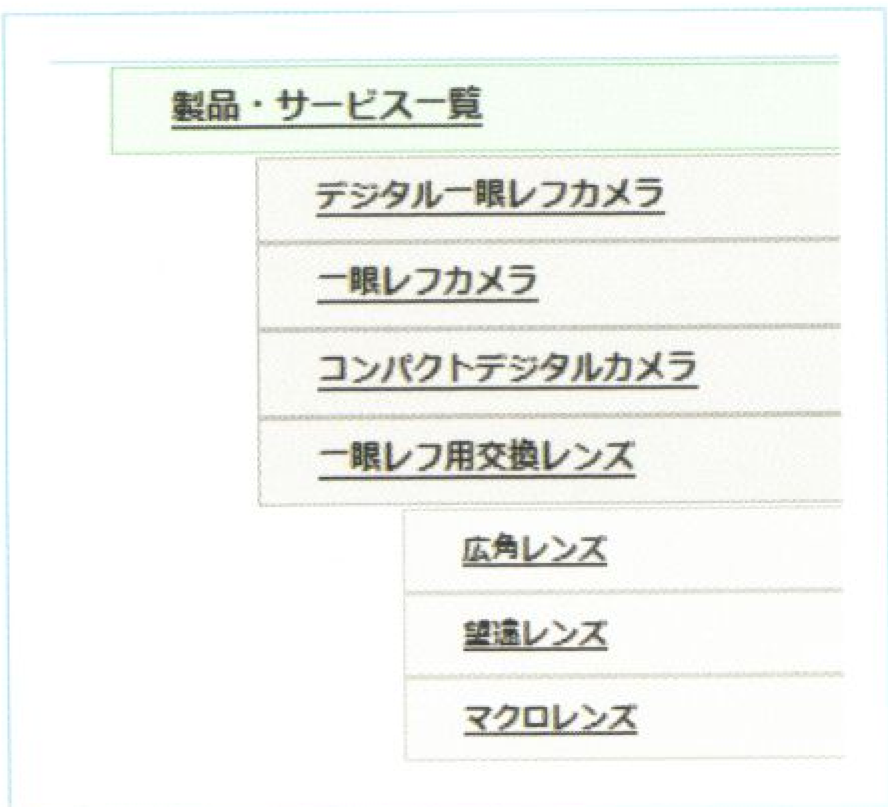
ツリー図の子要素となるulすべての左側に50pxの余白（margin）を空けることで、階層構造をわかりやすく表現しました。



TECHNIC 2

CSSの上書きでクロスブラウザに対応

IEでのレイアウトの崩れを防ぎ、最低限の表示を確保できるように、CSSを上書きすることで対応しています（左の画面が適切な表示、対して右の画面は対策を施したIE7の表示）。





ベースのHTMLを作成する

01

サンプルはHTML5で作成しています。IE8以下の古いブラウザにもHTML5の要素を適用させるため、「html5.js」というJavaScriptファイルを読み込みます 01-1。

「html5.js」をここでは「scripts」フォルダに格納し、index.htmlに 01-2 のように記述します。

<!--[if lte IE 8]> ～ <![endif]--> で囲っているのは条件分岐コメントで、HTML5に対応していないIE8以下でhtml5.jsのファイルを読み込むように指示しているものです。

01-1 html5.js

http://html5shim.googlecode.com/svn/trunk/html5.js

01-2 html5.jsの読み込み（index.html）

```
<!--[if lte IE 8]>
<script src="./scripts/html5.js" type="text/javascript"></script>
<![endif]-->
```

MEMO

「html5.js」については、INTRODUCTIONの「CSS3使用時の設計指針と注意点」（16ページ）もご覧ください。

02

次に、ブラウザによってCSSの各要素のデフォルト値が違うのを揃えるために、「html5reset-1.6.1.css」というファイルを読み込みます 02-1。HTMLファイルでは 02-2 のように記述しました。

02-1 html5reset-1.6.1.css

http://code.google.com/p/html5resetcss/downloads/detail?name=html5reset-1.6.1.css&can=2&q=

02-2 CSSの読み込み（index.html）

```
<link rel="stylesheet" href="./styles/html5reset-1.6.1.css" type="text/css" />
<link rel="stylesheet" href="./styles/default.css" type="text/css" />
<link rel="stylesheet" href="./styles/content.css" type="text/css" />
<!--[if lte IE 9]>
<link rel="stylesheet" href="./styles/ie6_7_8_9.css" type="text/css" />
<![endif]-->
<!--[if lte IE 6]>
<link rel="stylesheet" href="./styles/ie6.css" type="text/css" />
<![endif]-->
```

MEMO

「default.css」はページ全体のレイアウトの大枠について記述したCSSファイルです。「content.css」にはツリー図に適用するスタイルをまとめました。これらのCSSを「styles」フォルダに格納して読み込ませています。「ie6_7_8_9.css」と「ie6.css」については後述します。

03

ツリー図のリストのマークアップはul要素を入れ子にするだけです。

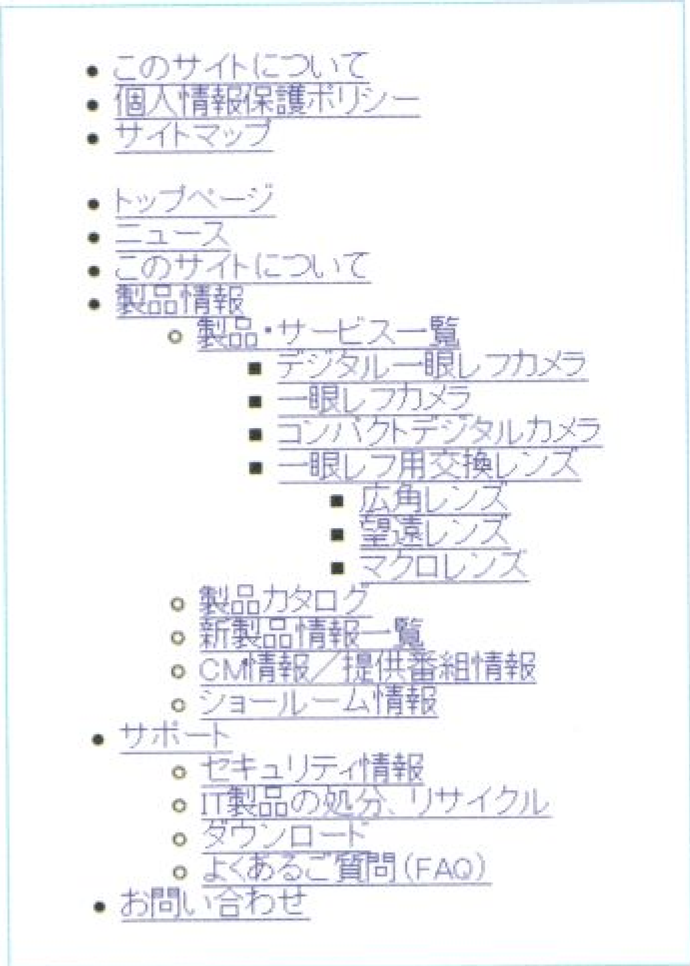
ここでは右上に表示されるリストはidをnav01として、その下に表示させたいリストはidをnav02として指定しています 03-1。

今回のサンプルでは4階層目までのCSSを指定しています。それよりも深くなる場合はCSSの記述を追加してください。

03-1 リストの階層（index.html）

```
<section id="demoArea">
  <article>
    <ul id="nav01">
      <li><a href="/terms/">このサイトについて
      </a></li>
      <li><a href="/privacy/">個人情報保護ポリシー
      </a></li>
      <li><a href="/sitemap/">サイトマップ</a>
      </li>
    </ul>
    <ul id="nav02">
      <li><a href="/">トップページ</a></li>
      <li><a href="/news/">ニュース</a></li>
      <li><a href="/about/">このサイトについて
      </a></li>
      <li><a href="/products/">製品情報</a>
      ― 省略 ―
    </ul>
  </article>
</section><!-- /demoArea -->
```

03-2 ブラウザで表示させたところ

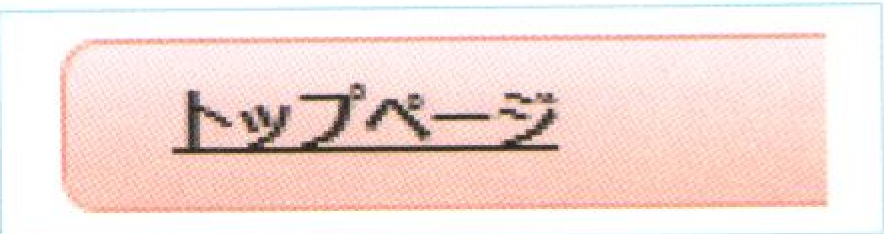


リストを装飾して階層を表現する

04

リストをCSS3で装飾します。IEでは確認することができませんが、今回のサンプルではbackgroundプロパティのlinear-gradientで背景はグラデーションさせ、border-radiusプロパティで角丸の表現も加えています 04-1 04-2。

04-2 リストに背景のグラデーションと角丸の表現を加えた



04-1 背景のグラデーションと角丸の表現（content.css）

```
#demoArea ul#nav01 li a {
  display: block;
  padding: 8px 20px 25px;
  color: #000;
  border: 1px solid #ffcc7f;
  background: -moz-linear-gradient(top, #fff5e5, #ffe062);
  background: -webkit-gradient(linear, left top, left bottom, from(#fea), to(#ffe062));
  background: linear-gradient(top, #fea, #ffe062);
  -webkit-border-radius: 7px;
  -moz-border-radius: 7px;
  border-radius: 7px;
}
```

MEMO

backgroundプロパティのlinear-gradientで注意すべき点は、
FirefoxとWebKit系ブラウザ（Chrome、Safari）とで指定のしかたが違う点です 01。

01 backgroundプロパティのlinear-gradientの構文

```
background: -moz-linear-gradient(開始位置と角度, 開始色, 途中色, 終了色 );
background: -webkit-gradient(linear, 開始位置, 終了位置 , from(開始色 ), color-stop(位置, 途中色 ), to(終了色 ));
```



05

リストの階層構造は、ul#nav02の子要素となるulすべての左側に50pxの余白を設けることで、階層を見た目にわかりやすく表現しています 05-1 05-2 。

05-1 余白による階層表現（content.css）

```
#demoArea ul#nav02 ul {
  margin: 1px 0 0 50px;
}
```

05-2 左側に余白を空けて階層を表現している



06

06-1 のようにul要素を増やしていくことで、深い階層の表現が可能となります。

06-1 ul要素を増やすことで深い階層を指定（content.css）

```
#demoArea ul#nav02 li a {
  ― 省略 ―
}
#demoArea ul#nav02 ul li a {
  ― 省略 ―
}
```

```
#demoArea ul#nav02 ul ul li a {
  ― 省略 ―
}
#demoArea ul#nav02 ul ul ul li a {
  ― 省略 ―
}
```

IEでの表示に備えた対応

07

IEで表示した場合でも情報を読み取ることができるよう、対策を施していきます。
まずは背景色の設定です。IEではbackgroundプロパティのlinear-gradientが効かないため、背景色が指定されていない状態に上書きして色指定をしています 07-1 。

07-1 IE用に背景色を上書きする指定（ie6_7_8_9.css）

```
#demoArea ul#nav01 li a {
  background: #fff5e5;
}
#demoArea ul#nav02 > li:first-child a {
  background: #ffeaef;
}
#demoArea ul#nav02 li a {
  background: #e5f5fa;
}
#demoArea ul#nav02 ul li a {
  background: #e5f5ef;
}
#demoArea ul#nav02 ul ul li a {
  background: #efefef;
}
#demoArea ul#nav02 ul ul ul li a {
  background: #f5f5f5;
}
```

MEMO

:first-child擬似クラスは、最初に現れる子要素にスタイルを適用する働きをします。

08

IE6のli要素の中のaにdisplay: block;とすると余白ができるため、li要素にdisplay: inline;を指定しました 08-1。

5階層、6階層と深くなる場合は、前のステップで紹介してきたcontent.cssにul要素を増やします。

08-1 IE6用の指定（ie6.css）

```
#demoArea ul#nav02 li,
#demoArea ul#nav02 ul li,
#demoArea ul#nav02 ul ul li,
#demoArea ul#nav02 ul ul ul li { display:
inline; }
```

リンク先を表示させるjsファイル

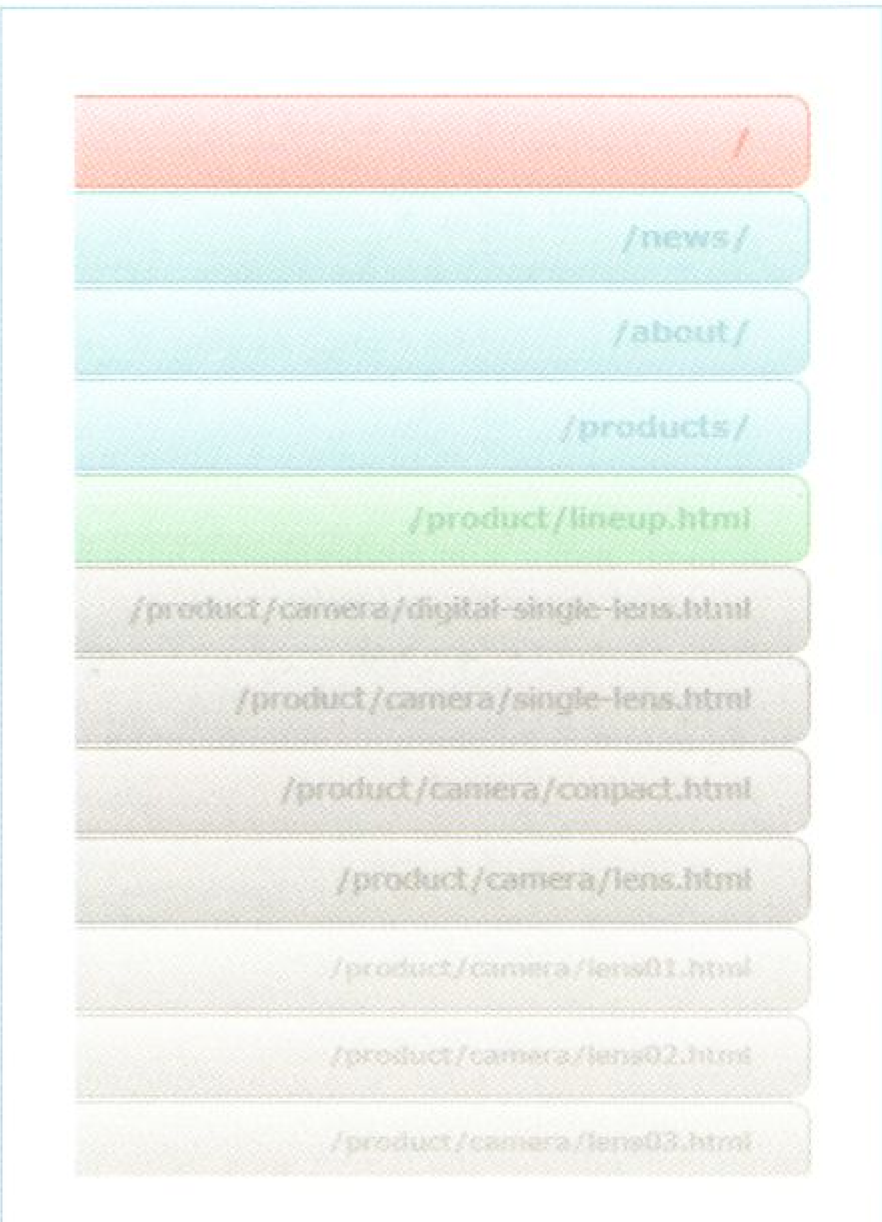
09

このサンプルでは、09-1のようにリンク先を表示させるJavaScript（tree-graph.js）も作成しました。

tree-graph.js 09-2では、li要素の中のa要素のhrefを取得し、span要素として追加しています。使い方はHTMLのhead内に記述し、tree-graph.jsを読み込むだけです 09-3。

（tree-graph.jsは筆者が作成したものですので、ご自由にお使いください。）

09-1 リンク先の表示（CSS適用後）



09-2 tree-graph.js

```
jQuery(function(){
  $('#nav01 li').each(function() {
    var linkHref = $(this).children('a').
    attr('href');
    $(this).append('<span>' + linkHref + '</
    span>');
  });
  $('#nav02 li').each(function() {
    var linkHref = $(this).children('a').
    attr('href');
    $(this).append('<span>' + linkHref + '</
    span>');
  });
});
```

09-3 jQueryとtree-graph.jsの読み込み（index.html）

```
<head>
― 省略 ―
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.
min.js"></script>
<script src="./scripts/tree-graph.js" type="text/javascript"></script>
― 省略 ―
</head>
```

まとめ [CONCLUSION]

今回はプログレッシブ・エンハンスメントの考え方でサンプル作成を行いました。プログレッシブ・エンハンスメントとは「どのユーザーにも同じように情報コンテンツを提供する。その中で環境が整っているユーザー（モダンブラウザ利用者）には、よりリッチな見せ方でコンテンツを提供する」という考え方です。こういった考え方に沿ってコーディング作業を行うと、作業にムダが少なくなり効率が上がります。

その他、マークアップするだけでいろいろな表現が可能になるCSSライブラリも多く存在します。気になったものから少しずつ試してCSS3に慣れるといいでしょう。

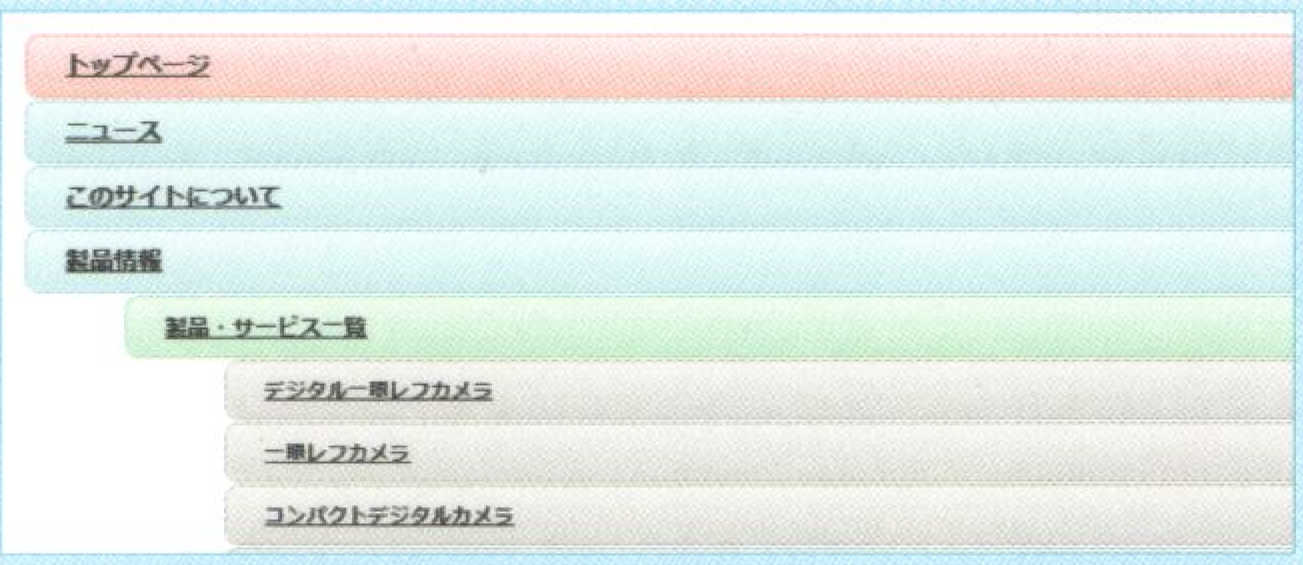


*COLUMN

IE対策と表示の違い

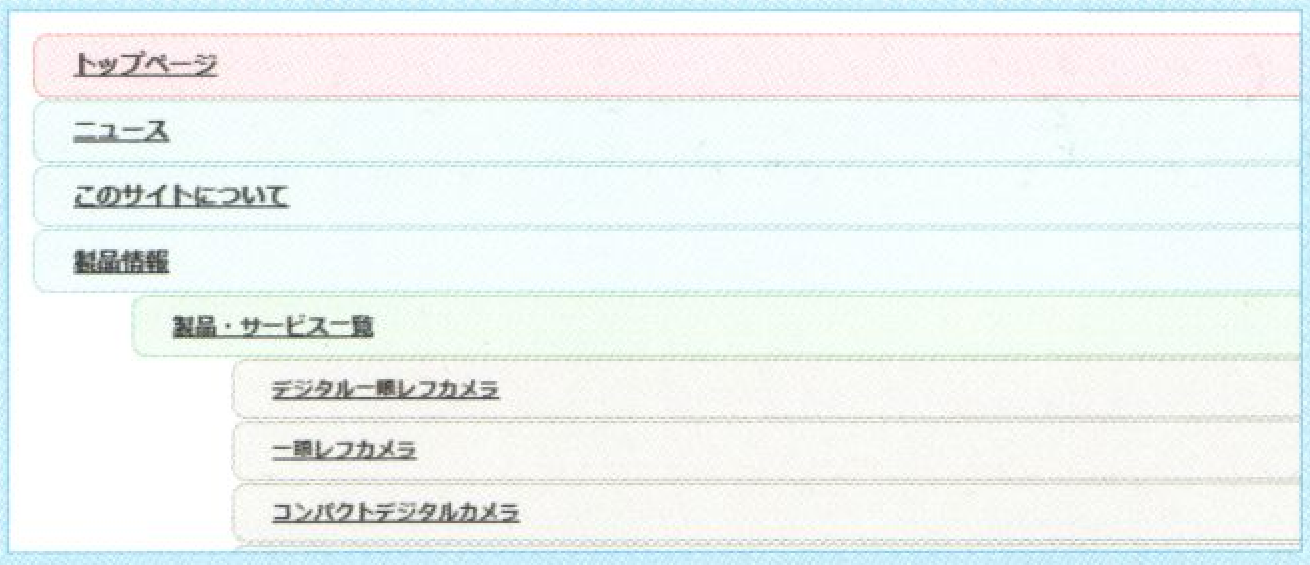
ここでは、IEで表示した際にレイアウトが崩れるのを防いだり、情報が読み取れるような最低限の表示を確保したりするために、ie6_7_8_9.css（IE用）とie6.css（IE6用）のファイルを作成して、CSSを上書きすることでクロスブラウザに対応しました。実際に表示がどのように違うかは、01のキャプチャ画像で確認してください。

01 **ブラウザによる表示の違い／ Firefox**



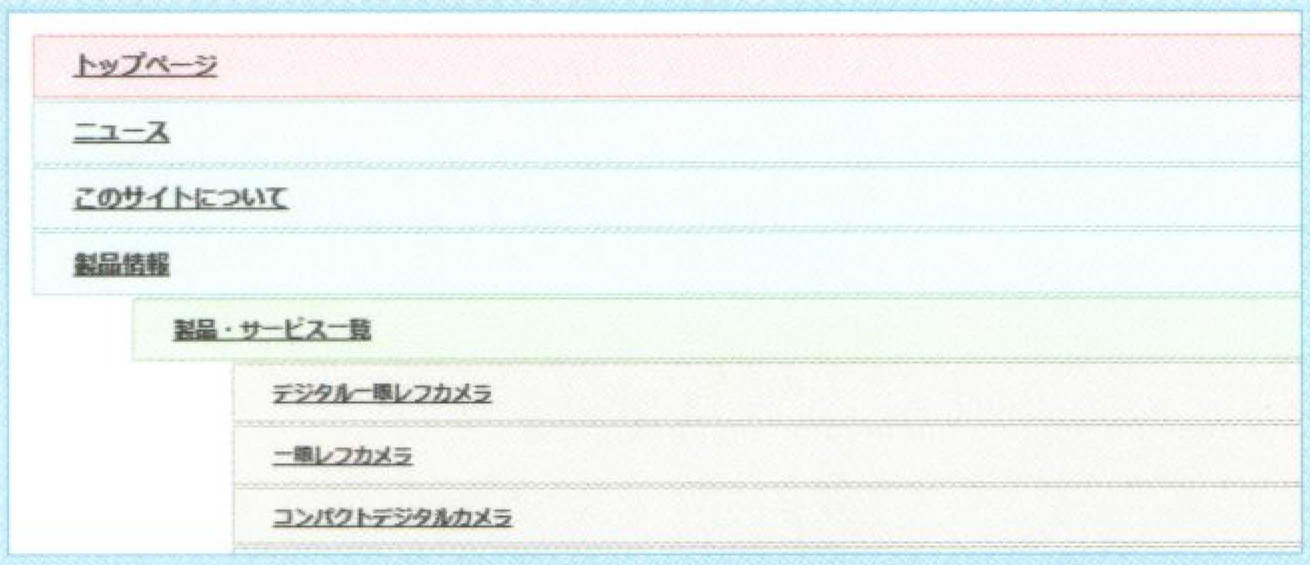
指定したCSSがすべて反映されている

IE9



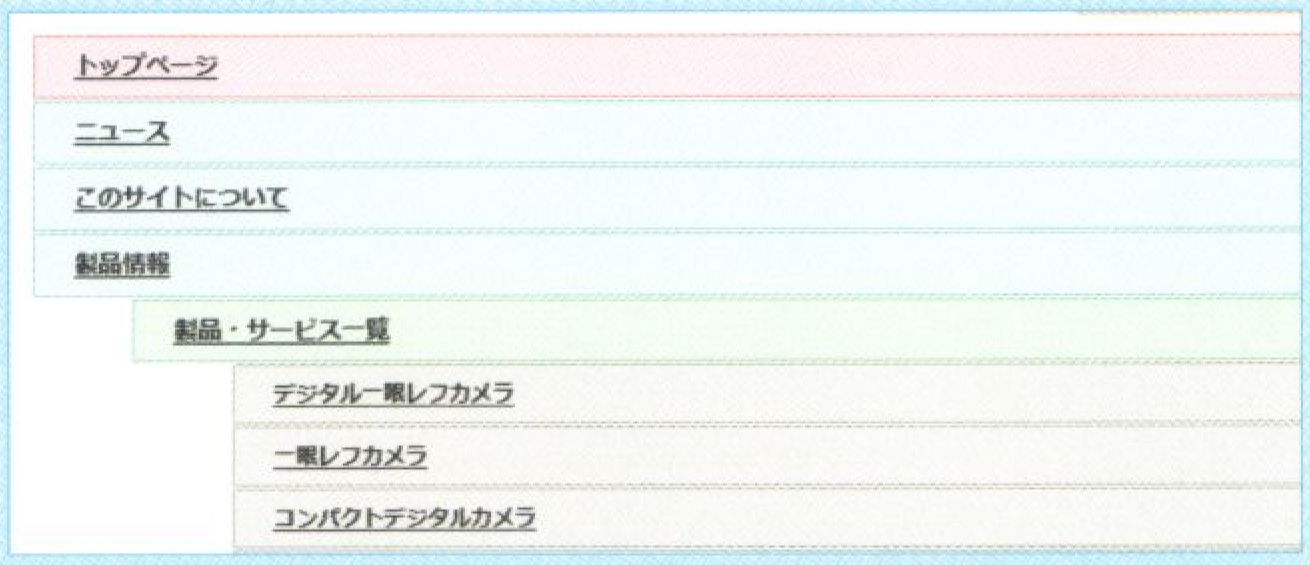
線形グラデーションを指定するlinear-gradientが反映されていない

IE8



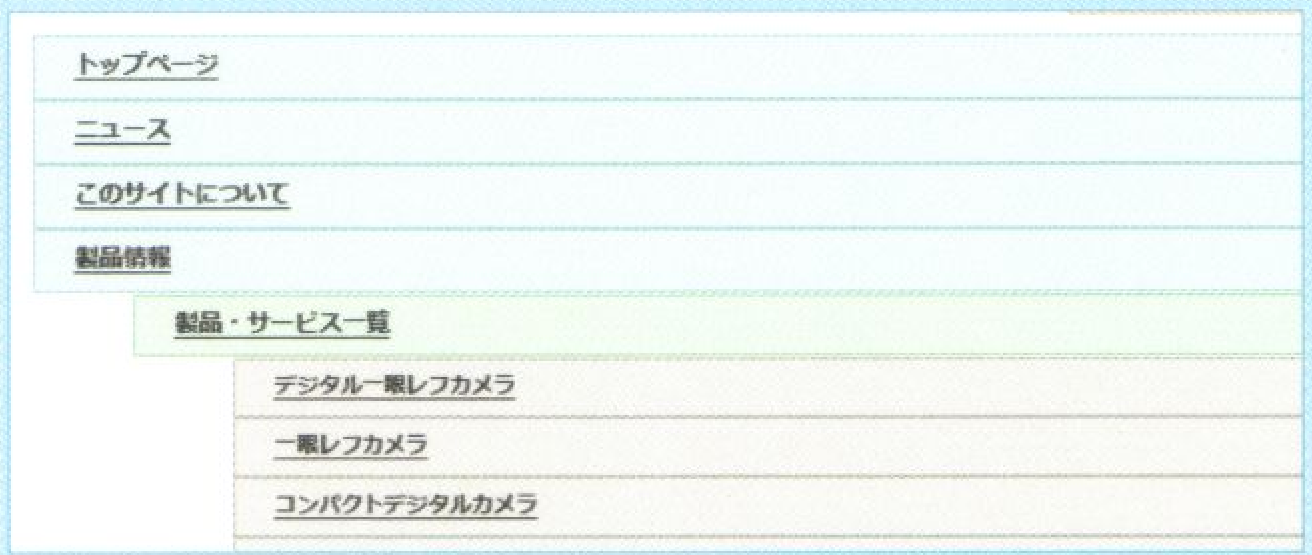
線形グラデーションもborder-radiusプロパティの角丸も反映されていない

IE7



線形グラデーションもborder-radiusプロパティの角丸も反映されていない。今回のサンプルだと見え方はIE8とよく似ている

IE6



線形グラデーションも角丸も反映されていない。また、最初に現れる子要素にスタイルを適用する:first-child擬似クラスでの指定も反映されていないため、「トップページ」部分が赤色の背景になっていない

このように反映されていない箇所はありますが、ツリー図としての機能や見え方は損なわないようにしました。
CSS3を導入する場合は、クロスブラウザ対応をどこまで実現するのか想定しないと思わぬ工数がかかるケースが少なくありません。

05 borderと背景の組み合わせやoutlineで実現する飾り罫線



CSS3では、背景画像を複数指定して組み合わせることで、より複雑な背景を簡単に実現することができます。
ボーダーやアウトラインもアルファ値の指定などと組み合わせることにより、より凝った表現ができるようになりました。

制作・文／新谷剛史（株式会社セカンドファクトリー）

制作のポイント

- アウトライン、ボーダー、背景を利用してボーダーに変化をつける
- 背景画像の始点が指定できることを有効活用する
- 幅や高さでのボーダーの取り扱いを理解する

対応ブラウザ

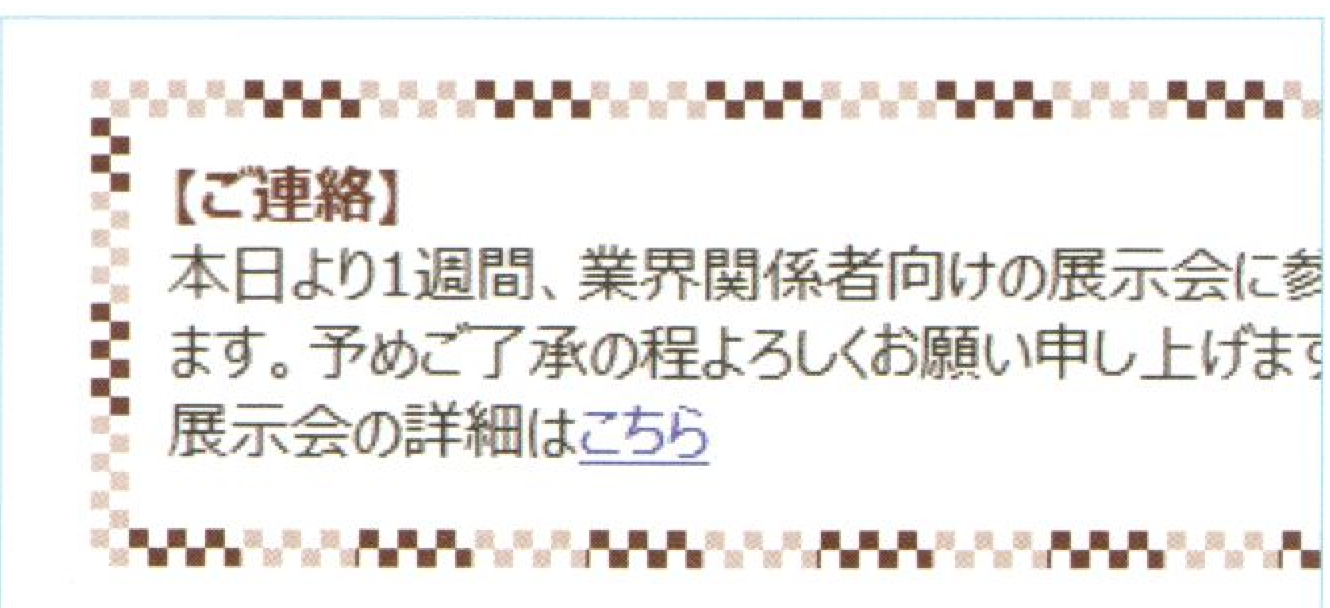
IE	9 以上
Safari	5 以上
Firefox	3.6 以上
Chrome	10 以上

※IEでは一部に非対応。

TECHNIC 1

背景画像の始点とボーダー

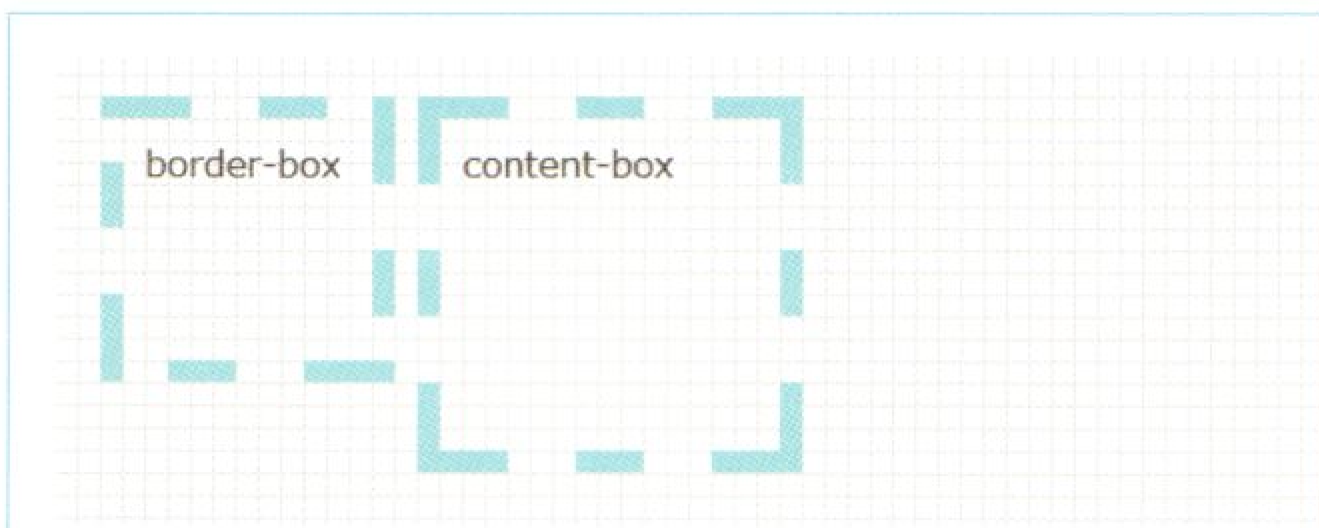
従来、背景画像の始点はコンテンツを基準にしていましたが、ボーダーの左上を背景画像の始点にできるようになりました。



TECHNIC 2

幅・高さとボーダー

floatやボックスレイアウトによりマルチカラムレイアウトを実現する場合、ボーダーの扱いが少々面倒でしたが、ボーダーを幅や高さに含むborder-box指定ができるようになりました。





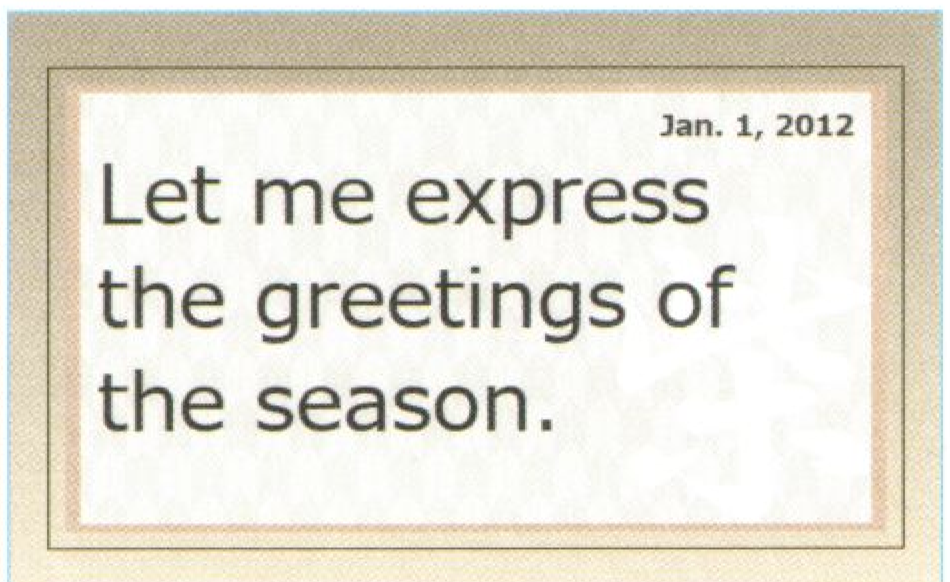
アウトラインとボーダー

01

CSSでは、ボーダーのほかに要素の輪郭を表す線をoutlineプロパティで指定できます。CSS3から、新たにボーダーとアウトラインの間隔を調整するoutline-offsetプロパティが用意されました。

outline-offsetを指定すると、01-1のようにボーダーとアウトラインの間に隙間を設定することができます01-2 01-3。この隙間で見える背景は、outlineを指定した要素の親要素の背景です。

01-1 ボーダーとアウトラインの間に隙間を作る（sample02.html）



01-2 アウトラインとボーダー（sample02.html）

```
<div class="thcont">
  <h2>Top NEWS <span>Jan. 1, 2013</span></h2>
  <p>Let me express the greetings of the season.</p>
</div>
```

01-3 outline-offsetの指定（sample02のCSS）

```
div.thcont{
position:absolute; top:102px; left:427px;
width:480px; height:270px;
background:url(../img/bg03.png) no-repeat bottom right,
url(../img/bg02.png) repeat top left;
border-top:#E6C7AE 5px solid;
border-right:#E6C7AE 9px solid;
border-bottom:#E6C7AE 5px solid;
border-left:#E6C7AE 9px solid;
outline:#371C09 solid 1px;
outline-offset:10px;
}
```

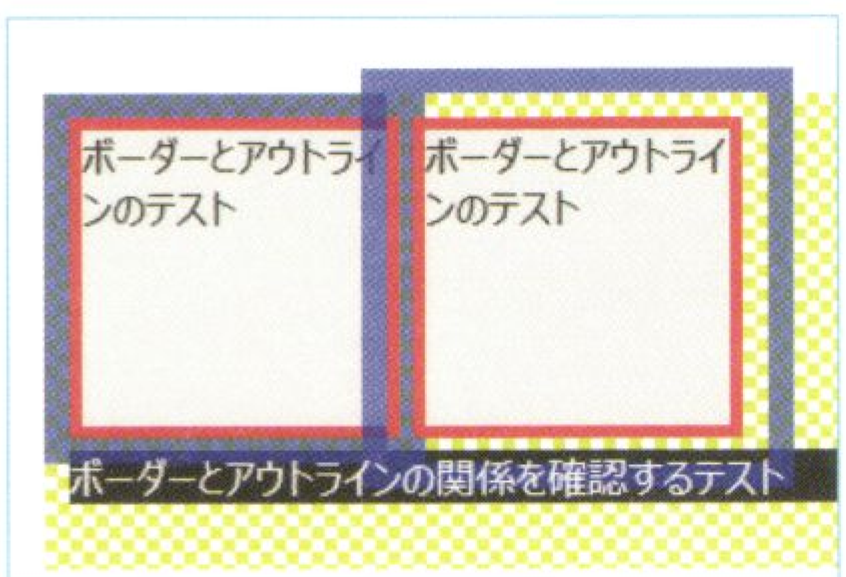
02

outlineは輪郭を表すための線ですので、ボーダーのように自由な設定はできません。幅・高さなどには考慮されない点に注意しましょう。

02-1では、右と下にマージンを5pxとった2つのボックスを配置しています。アウトライン、ボーダー、マージンと、背景が黒い確認用ブロック要素との関係を把握しましょう02-2 02-3。

なお、アウトラインの色には、CSS3から指定可能になったアルファ値が利用可能です。

02-1 ボーダーとアウトラインの関係（sample02.html）



02-3 斜め方向へのグラデーション（sample02のCSS）

```
body.top div.content .topbox01,
body.top div.content .topbox02{
float:left;
width:120px; height:120px;
margin:0 5px 5px 0;
background:#F0F0F0;
border:#F33 solid 5px;
outline:rgba(0,64,198,0.7) solid 10px;
}
body.top div.content .topbox02{
outline-offset:10px;
}
```

02-2 HTMLの記述（sample02.html）

```
<div class="topbox01wrap">
  <p class="topbox01">ボーダーとアウトラインのテスト</p>
  <p class="topbox02">ボーダーとアウトラインのテスト</p>
  <p class="checktext">ボーダーとアウトラインの関係を確認するテスト</p>
</div>
```

画像やブロック要素と組み合わせるボーダー

03

ボーダーはCSS2とほとんど一緒ですが、ボーダー色にアルファ値が指定できるようになりました 03-1。
ここではボーダー部分の白にアルファ値を指定し、アルファ値が有効になっていることを確認するため、背景色を明るいブルーにしました 03-2 03-3。

03-1 ボーダーとアウトラインの関係



03-2 HTMLの記述（sample03.html）

```
<p class="border01"><img alt="Cherry
blossom" src="img/spring.jpg" /></p>
```

03-3 ボーダー部分の白にアルファ値を指定（sample03のCSS）

```
p.border01 img{
display:inline-block;
-moz-box-shadow:rgba(44,39,39,0.3) 3px 3px 7px 7px;
-webkit-box-shadow:rgba(44,39,39,0.3) 3px 3px 7px 7px;
box-shadow:rgba(44,39,39,0.3) 3px 3px 7px 7px;
-ms-transform:rotate(-3deg);
-moz-transform:rotate(-3deg);
-webkit-transform:rotate(-3deg);
-o-transform:rotate(-3deg);
transform:rotate(-3deg);
background:#5FD5F3;
border:rgba(255,255,255,0.8) 10px solid;
}
```

背景画像とボーダーを組み合わせる

04

ボーダーにアルファ値を指定し、背景画像を組み合わせると、この例のように従来のボーダーでは難しかったボーダーの表現ができます 04-1。ここでは、シンプルな市松模様の背景画像と組み合わせましたが、より凝った表現も実現できます。
ボーダーと背景はdivタグ、コンテンツ部分の白い背景はpタグで実現しています 04-2 04-3。

04-1 ボーダーと透過を組み合わせた例（sample03.html）



04-2 HTMLの記述（sample03.html）

```
<div class="topbox03"><p>ボーダーと透
過を組み合わせると、こんな変化も実現で
きます。</p></div>
```

04-3 CSSの記述（sample03）

```
div.topbox03{
background:url(../img/bg04.png);
border:rgba(95,213,243,0.6) dashed 10px;
}
```

05

背景画像とボーダーによる表現は、実はdivタグ＋pタグのような2つのブロック要素を利用せず、1つのブロック要素だけでも実現できます。
背景画像は、CSS3で複数指定できるようになったので、リピート方向と背景画像の始点を設定して四辺のみに配置されるようにします。ボーダーの指定は 04 と一緒です 05-1 05-2。ただし、これだけではボーダーの裏に背景画像が配置されません 05-3。
背景はボーダー領域の内側に配置されます。背景の始点はボーダーの内側、padding領域の左上ですから、ほとんどのボーダー領域に背景が表示されない状態です。そのため、背景の始点をボーダー領域の左上にします。background-originプロパティの値をborder-boxにすると、ボーダー領域の左上が始点になります 05-4。



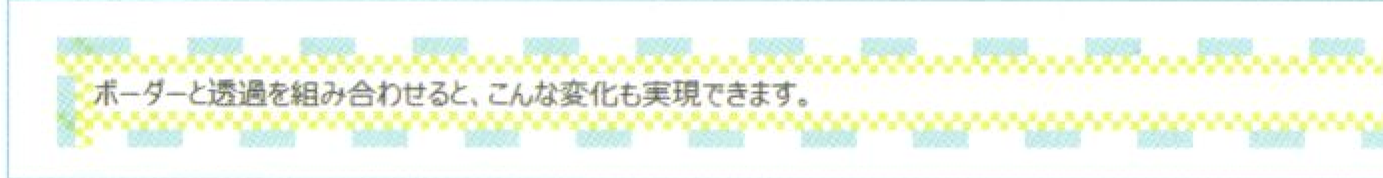
05-1 HTMLの記述（sample03.html）

```
<p class="topbox04">ボーダーと透過を組み合わせると、こんな変化も実現できます。</p>
```

05-2 背景画像とボーダーの組み合わせ（途中／sample03のCSS）

```
p.topbox04b{
padding:10px;
background:url(../img/bg04.png) repeat-x top left,
url(../img/bg04.png) repeat-y top right,
url(../img/bg04.png) repeat-x bottom left,
url(../img/bg04.png) repeat-y top left;
border:rgba(95,213,243,0.6) dashed 10px;
}
```

05-3 ボーダー領域に背景が表示されない（sample03.html）



05-4 背景画像とボーダーの組み合わせ（sample03のCSS）

```
p.topbox04{
padding:10px;
background:url(../img/bg04.png) repeat-x top left,
url(../img/bg04.png) repeat-y top right,
url(../img/bg04.png) repeat-x bottom left,
url(../img/bg04.png) repeat-y top left;
background-origin:border-box;
border:rgba(95,213,243,0.6) dashed 10px;
}
```

06

背景画像とボーダーに関する設定は、05で紹介したbackground-originのほかにも、背景を配置する範囲を定義できるbackground-clipや、box-sizingがあります。
06-1はbackground-clipで指定する値を変えたときの表示例です06-2。
border-boxは幅、高さの値をボックス領域まで含んで適用する、ブラウザの互換モードのような算出方法です。
content-boxはCSS2.1の定義に沿った算出方法で、ボーダーは幅や高さに含みません06-3 06-4。

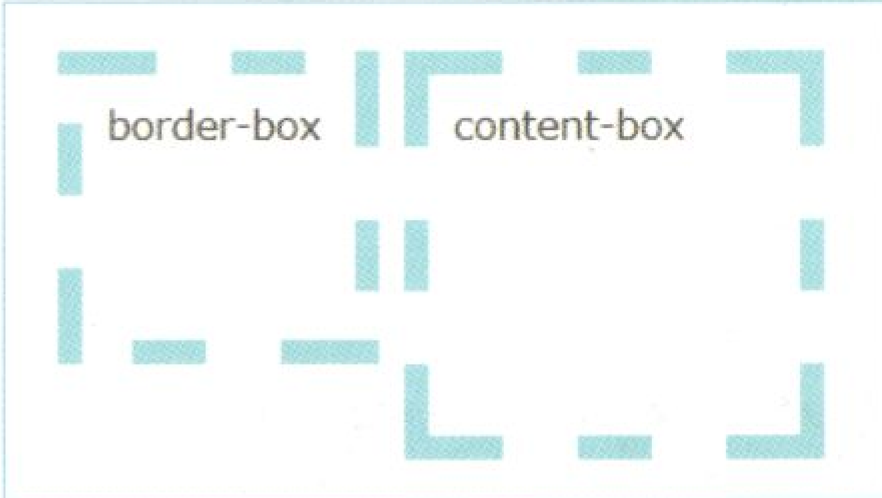
06-1 background-clipを使ったボーダーの例（sample03.html）



06-2 border-box、padding-box、content-boxを指定（sample03のCSS）

```
p.topbox04c-b{
background-clip:border-box;
}
p.topbox04c-p{
background-clip:padding-box;
}
p.topbox04c-c{
background-clip:content-box;
}
```

06-3 border-boxとcontent-box（sample03.html）



06-4 border-boxを指定（sample03のCSS）

```
p.topbox05-b{
-moz-box-sizing:border-box;
-webkit-box-sizing:border-box;
box-sizing:border-box;
}
p.topbox05-c{
-moz-box-sizing:content-box;
-webkit-box-sizing:content-box;
box-sizing:content-box;
}
```

まとめ [CONCLUSION]

アウトラインを利用する例はそれほど多くはないかもしれませんが、どのようなことができるのかという点については押さえておくとよいでしょう。
ボーダーについては、CSS3で線の色に透過色が利用できるようになりましたが、それほど大きな変更はありません。しかし、従来のボーダーでは実現できなかった表現が可能になったことで、背景画像の重ね合わせなどのテクニックに頼ることなく実現できる表現が増えました。
今後よく使うテクニックのひとつになると思いますので、内容をしっかりと把握しましょう。

06 グラデーションと角丸で作る見出しのバリエーション





CSS3ではborder-radiusプロパティが新たに追加されたほか、backgroundプロパティなどの値にグラデーションを指定することができるようになりました。これらを利用することで、複雑なコードでしか実現できなかった表現が、シンプルなコードで実現可能になります。

制作・文／新谷剛史（株式会社セカンドファクトリー）

制作のポイント

- ブラウザごとに差のある背景グラデーションの表記方法を押さえる
- 併用可能な新しいプロパティを活用
- 角丸の表記法を押さえる

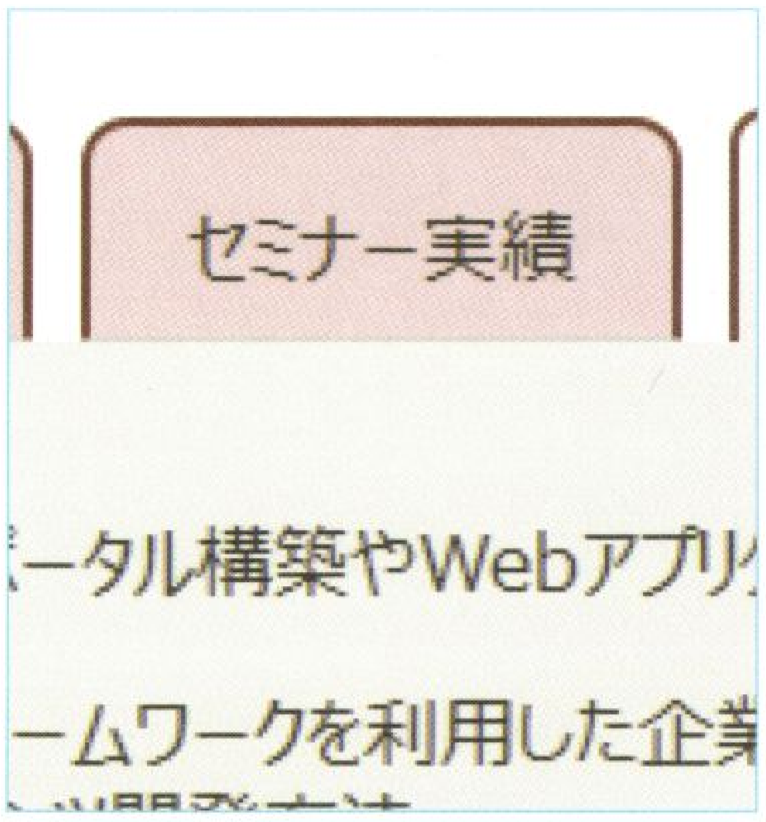
対応ブラウザ

ブラウザ	対応
IE	非対応
Safari	5 以上
Firefox	3.6 以上
Chrome	10 以上

TECHNIC 1

角丸利用時にはpaddingに注意

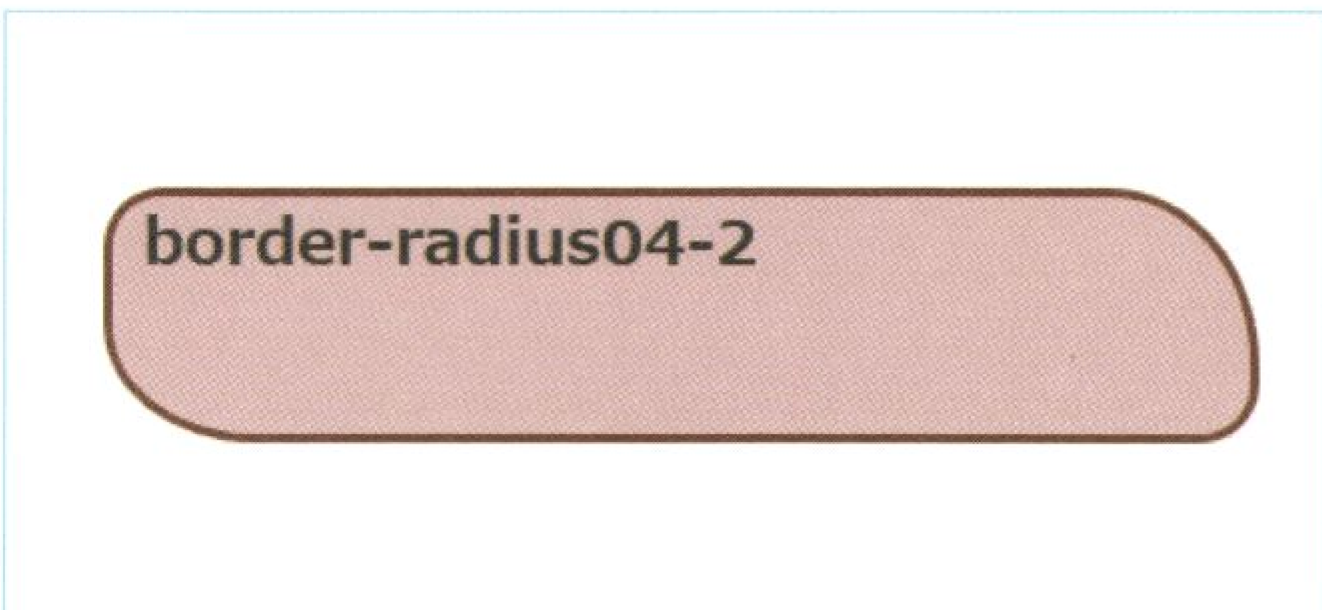
文字と角丸が重なることを防ぐため、通常左右のpaddingを、角丸半径と同等かそれ以上の値にします。



TECHNIC 2

楕円の角丸も可能

角丸半径の縦方向、横方向の半径をそれぞれ設定することにより、楕円の角丸を実現することも可能です。





いろいろなグラデーションの利用

01

CSS3では、線形と放射状（円形）のグラデーションが設定できます01-1。指定する方法は、ブラウザによって若干の違いがありますので、注意が必要です。Firefox系のブラウザはCSS3の書式と同じ書式ですが、ベンダープレフィックス-moz-をつけます。SafariやChrome系ブラウザなど、ベンダープレフィックスが-webkit-のものは、CSS3と異なる書式です。

01-2は縦方向へのグラデーションです。書式は異なりますが、横方向の指定01-3と比較すると、理解は容易なはずです。

書式の違いにより、斜め方向へのグラデーションは大きく異なります。CSS3の書式では角度を指定しますが、-webkit-では角度ではなく「left top,right bottom」のように始点と終点を指定して斜め方向を指定します01-4。

01-1 縦、横、斜めのグラデーション（sample01.html）



01-2 縦方向のグラデーション（sample01のCSS）

```
h3.gradient01-01{
background:-moz-linear-gradient(#5FD5F3,#A7B311);
background:-webkit-gradient(linear,left top,left bottom,from(#5FD5F3),to(#A7B311));
background:linear-gradient(#5FD5F3,#A7B311);
}
```

01-3 横方向のグラデーション（sample01のCSS）

```
h3.gradient01-02{
background:-moz-linear-gradient(right,#5FD5F3,#A7B311);
background:-webkit-gradient(linear,right top,left top,from(#5FD5F3),to(#A7B311));
background:linear-gradient(right,#5FD5F3,#A7B311);
}
```

01-4 斜め方向へのグラデーション（sample01のCSS）

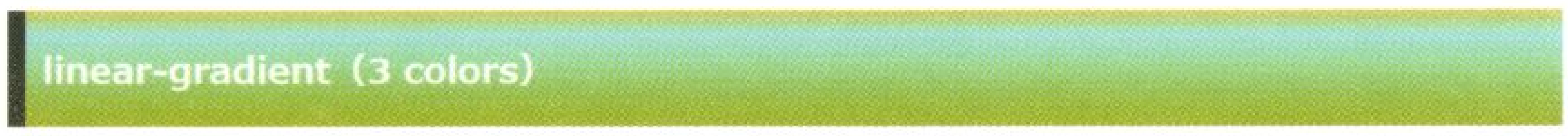
```
h3.gradient01-03{
background:-moz-linear-gradient(left bottom 45deg,#5FD5F3,#A7B311);
background:-webkit-gradient(linear,left bottom,right top,from(#5FD5F3),to(#A7B311));
background:linear-gradient(left bottom 45deg,#5FD5F3,#A7B311);
}
```

02

3色以上のグラデーション指定を行うこともできます02-1。始点の色、終点の色、間の色と、始点からの距離を追加します02-2。

Firefox系では距離の指定にピクセルや％も使用できますが、SafariやChromeなど-webkit-ではピクセルで指定する距離が利用できないため、％で指定する必要があります。

02-1 3色のグラデーション（sample01.html）



02-2 3色以上のグラデーションを指定（sample01のCSS）

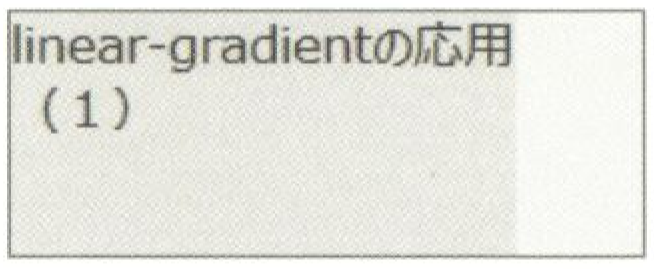
```
h3.gradient01-04{
background:-moz-linear-gradient(#EDCB46,#5FD5F3 15px,#A7B311);
background:-webkit-gradient(linear,left top,left bottom,from(#EDCB46),color-stop(20%,#5FD5F3),to(#A7B311));
background:linear-gradient(#EDCB46,#5FD5F3 15px,#A7B311);
}
```

グラデーションの応用

03

グラデーションの指定を応用すると、比較的複雑な背景もCSS3だけで実現することが可能です。
グラデーションの始点からある距離までを同じ色の指定、次に同じ距離でもう一度終点と同じ色の指定を行い、最後に終点を指定すると、始点から中間点までは同じ色なのでグラデーションなし、中間点から2つ目の中間点はグラデーションなのだが距離が一緒のため表示されず、最後に2つ目の中間点から終点も同じ色なので、結果としてグラデーションのない2色の塗りを実現することができます 03-1。
距離については後の作業を考え、%で指定しておきましょう 03-2。

03-1 グラデーションを組み合わせて作った2色の塗り（sample03.html）



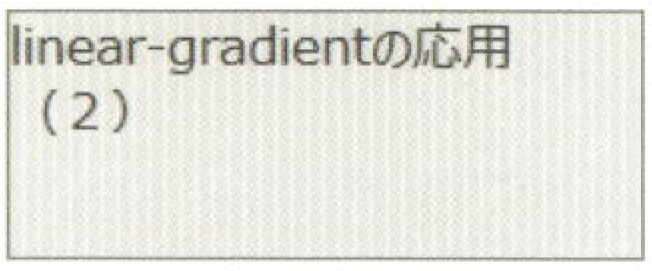
03-2 グラデーションの距離は%で指定する（sample03）

```
p.gradient02-01{
background:-moz-linear-gradient(left,#DDDDDD,#DDDDDD 80%,#EEEEEE 80%,#EEEEEE);
background:-webkit-gradient(linear,left top,right top,from(#DDDDDD),color-
stop(80%,#DDDDDD),color-stop(80%,#EEEEEE),to(#EEEEEE));
background:linear-gradient(left,#DDDDDD,#DDDDDD 80%,#EEEEEE 80%,#EEEEEE);
```

04

次に、このグラデーションを一定サイズでリピートさせましょう 04-1。
CSS3では、背景画像の幅と高さを指定できます。この指定は 03 のグラデーションでも有効です。
background-sizeの幅を5px、高さをautoで指定すると、縦じまの模様が表現できました 04-2。背景画像のリピートについては指定していなかったため、初期値のrepeatが有効になっています。

04-1 グラデーションで作った縦じま模様（sample03.html）



04-2 background-sizeの幅を指定する（sample03のCSS）

```
p.gradient02-02{
background:-moz-linear-gradient(left,#DDDDDD,#DDDDDD 80%,#EEEEEE 80%,#EEEEEE);
background:-webkit-gradient(linear,left top,right top,from(#DDDDDD),color-
stop(80%,#DDDDDD),color-stop(80%,#EEEEEE),to(#EEEEEE));
background:linear-gradient(left,#DDDDDD,#DDDDDD 80%,#EEEEEE 80%,#EEEEEE);
background-size:5px auto;
}
```

05

放射状のグラデーションも、基本的な指定方法はここまでの例と一緒です。
色の複数指定も可能です。
グラデーションの形状が円か楕円か、グラデーションのサイズを中心から近い辺や角にするのか、あるいは遠い辺や角にするのかなど、細かな指定もできます。
指定を組み合わせると、波紋のような表現も可能です 05-1 05-2。

05-1 放射状のグラデーション（sample04.html）



05-2 波紋のようなグラデーションを作る（sample04のCSS）

```
h3.gradient03{
background:-moz-radial-gradient(bottom left,circle farthest-side,#A7B311,#5FD5F3
20%,#81C588 23%,#5FD5F3 40%,#81C588 45%,#5FD5F3);
background:-webkit-gradient(radial,left bottom,0,left bottom,240,from(#A7B311),color-
stop(20%,#5FD5F3),color-stop(23%,#81C588),color-stop(40%,#5FD5F3),color-
stop(45%,#81C588),to(#5FD5F3));
background:radial-gradient(bottom left,circle farthest-side,#A7B311,#5FD5F3
20%,#81C588 23%,#5FD5F3 40%,#81C588 45%,#5FD5F3);
}
```



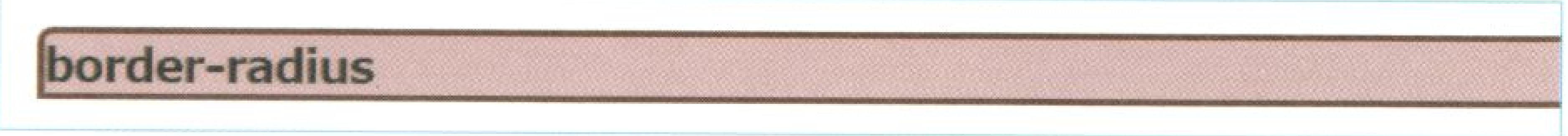
角丸で実現するタイトルやタブ

06

これまで、画像を利用しなければならなかった角丸も、CSS3では簡単に実現できるようになりました 06-1 。

古いバージョンのブラウザにも対応させるためには、ベンダープレフィックスのある指定とない指定を併記します。また、記述位置も若干異なるため、注意が必要です 06-2 。

06-1 左上が角丸になっている（sample04.html）



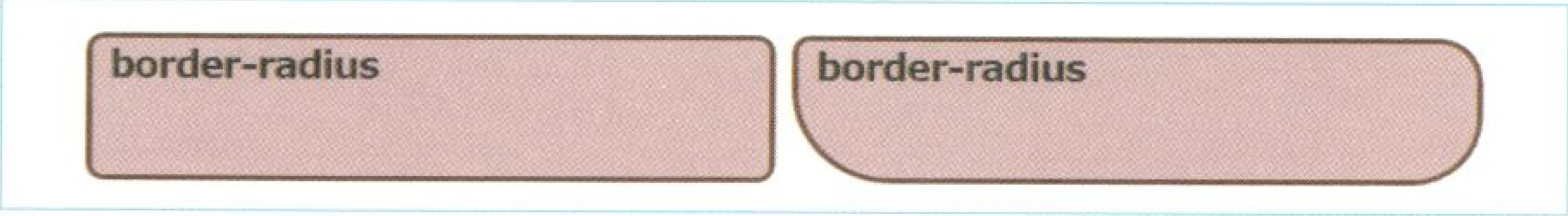
06-2 角丸の指定（sample04のCSS）

```
h3.borderRadius01{
border:#7D2A17 3px solid;
-moz-border-radius-topleft:7px;
-webkit-border-top-left-radius:7px;
border-top-left-radius:7px;
background:#DEB4B7
}
```

07

特定の角ではなく、すべての角を角丸にするためには、border-radiusまたはこれにベンダープレフィックスをつけるだけで指定できます 07-1 07-2 。また、同様の指定で半径を1つではなく4つ指定し、すべての角の半径を変更することも可能です。左上から時計回りに右上、右下、左下の順に指定したことになります 07-3 。

07-1 角丸の半径がすべて同じ（左）と、すべての角丸の半径が違う（右）（sample04.html）



07-2 すべての角を均等に丸くする（sample04のCSS）

```
h3.borderRadius02-1,
h3.borderRadius02-2{
float:left;
width:40%;
margin-left:10px;
background:#DEB4B7;
height:80px;
padding-left:10px;
border:#7D2A17 3px solid;
}
h3.borderRadius02-1{
-moz-border-radius:10px;
-webkit-border-radius:10px;
border-radius:10px;
}
```

07-3 半径が異なる指定(sample04のCSS)

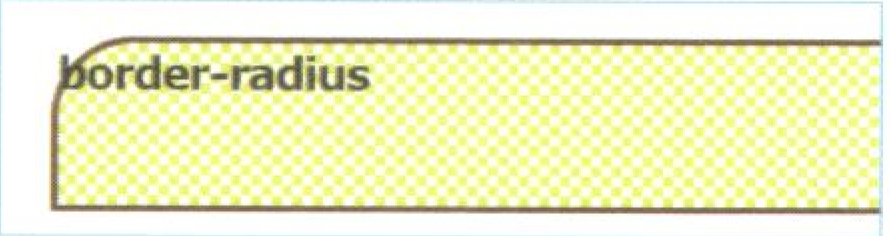
```
h3.borderRadius02-2{
-moz-border-radius:10px 25px 40px 55px;
-webkit-border-radius:10px 25px 40px 55px;
border-radius:10px 25px 40px 55px;
}
```

08

角丸を使うときは、少々注意が必要です。 08-1 では、左上の角丸半径を40ピクセルに指定しています。

背景はボーダーに合わせて表示されていますが、文字は角丸に無関係に配置されるため、ボーダーと重なって表示されてしまいます。
角丸を実現する場合は、左右か上下、どちらかのpaddingを半径と同程度以上に指定するのがよさそうです 08-2 。もちろん、半径以上の値であれば問題ありません。

08-1 文字が角丸からはみ出してしまう（sample04.html）



08-2 text-shadowの色指定で色を重ね合わせる（sample04のCSS）

```
h3.borderRadius03{
clear:both;
height:80px;
border:#7D2A17 3px solid;
-moz-border-radius-topleft:40px;
-webkit-border-top-left-radius:40px;
border-top-left-radius:40px;
background:url(../img/bg04.png);
}
```

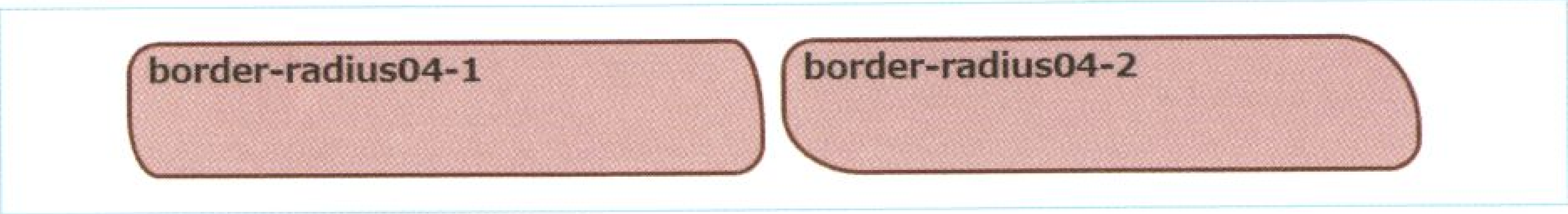
楕円の指定

09

楕円の指定を行うことも可能です。横方向の半径と縦方向の半径を「20px/50px」のように指定すると、縦長の楕円の一部が指定された状態になります 09-1。

複数の指定を行う場合、（横方向の半径指定）/（縦方向の半径指定）で、各方向の指定値が1つの場合はすべての半径、2つの場合は左上と右下、右上と左下の半径、3つの場合は左上、右上と左下、右下の半径、4つの場合は左上、右上、右下、左下それぞれの半径を指定したのと同じ状態です 09-2 09-3。

09-1 横方向の半径がすべて同じ場合（左）と、横方向の半径が異なる場合（右）（sample04.html）



09-2 横方向の指定値が1つ、縦方向の指定値が4つ（sample04のCSS）

```
h3.borderRadius04-1{
-moz-border-radius:20px / 20px 60px 20px 40px;
-webkit-border-radius:20px / 20px 60px 20px 40px;
border-radius:20px / 20px 60px 20px 40px;
}
```

09-3 横方向の指定値が2つ、縦方向の指定値が4つ（sample04のCSS）

```
h3.borderRadius04-2{
-moz-border-radius:20px 50px / 20px 60px 20px 40px;
-webkit-border-radius:20px 50px / 20px 60px 20px 40px;
border-radius:20px 50px / 20px 60px 20px 40px;
}
```

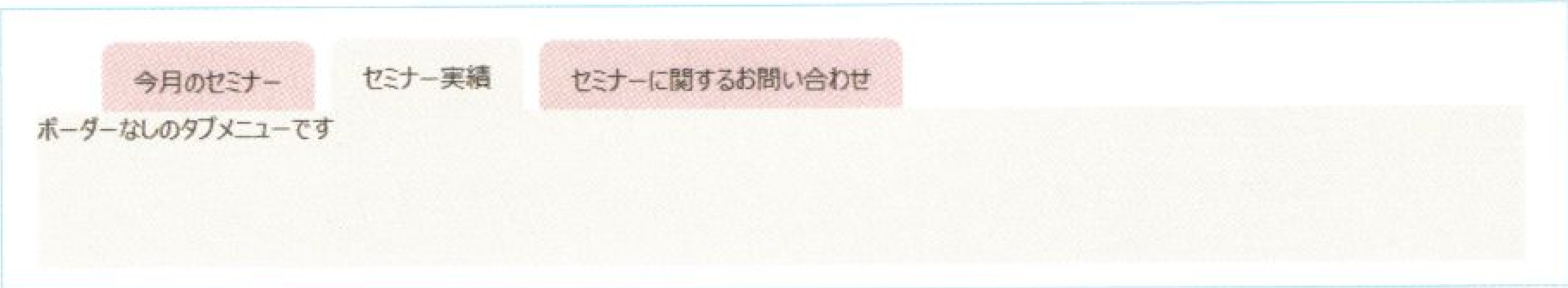
角丸をタブメニューに使う

10

角丸はタイトルだけではなく、タブメニューを実現する場合にも利用可能です 10-1。基本的な要領はタイトルの角丸と一緒です。

タブのアクティブ・非アクティブを実現するために背景色を変更したり、高さを少し変えるなどの表現を併用します 10-2。また、非アクティブなタブの下2つの角に角丸を少し入れておくと、それらしい雰囲気が出ます。

10-1 角丸で作ったタブメニュー（sample04.html）



10-2 タブのアクティブ・非アクティブを区別する（sample04のCSS）

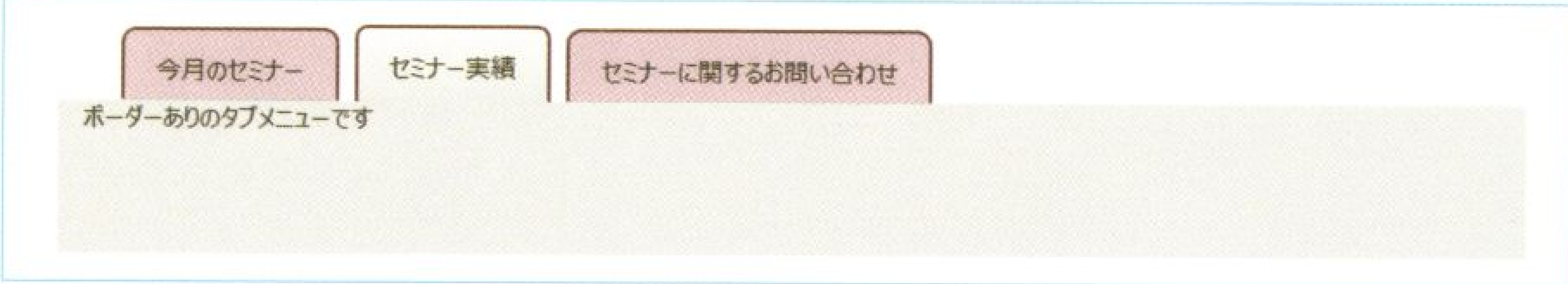
```
ul.tabMenu01 li{
float:left;
height:20px;
margin:2px 10px 0 0; padding:15px 20px 10px;
background:#ECD2D5;
-moz-border-radius:10px 10px 2px 2px;
-webkit-border-radius:10px 10px 2px 2px;
border-radius:10px 10px 2px 2px;
}
ul.tabMenu01 li.menuActive{
height:22px;
margin-top:0;
background:#EEE;
-moz-border-radius:8px 8px 0 0;
-webkit-border-radius:8px 8px 0 0;
border-radius:8px 8px 0 0;
}
```



11

ボーダーを併用する場合、下2つの角の角丸は入れないか、入れるとしても半径1px程度にとどめたほうが違和感がないように感じます。
代わりに非アクティブなタブにだけ、下にもボーダーを指定するとよいでしょう。この例では背景のグラデーションも指定しました 11-1 11-2 。
タブの角丸にも、楕円形などを指定できます。

11-1 ボーダーを追加したタブメニュー（sample04.html）



11-2 ボーダーとグラデーションの指定（sample04のCSS）

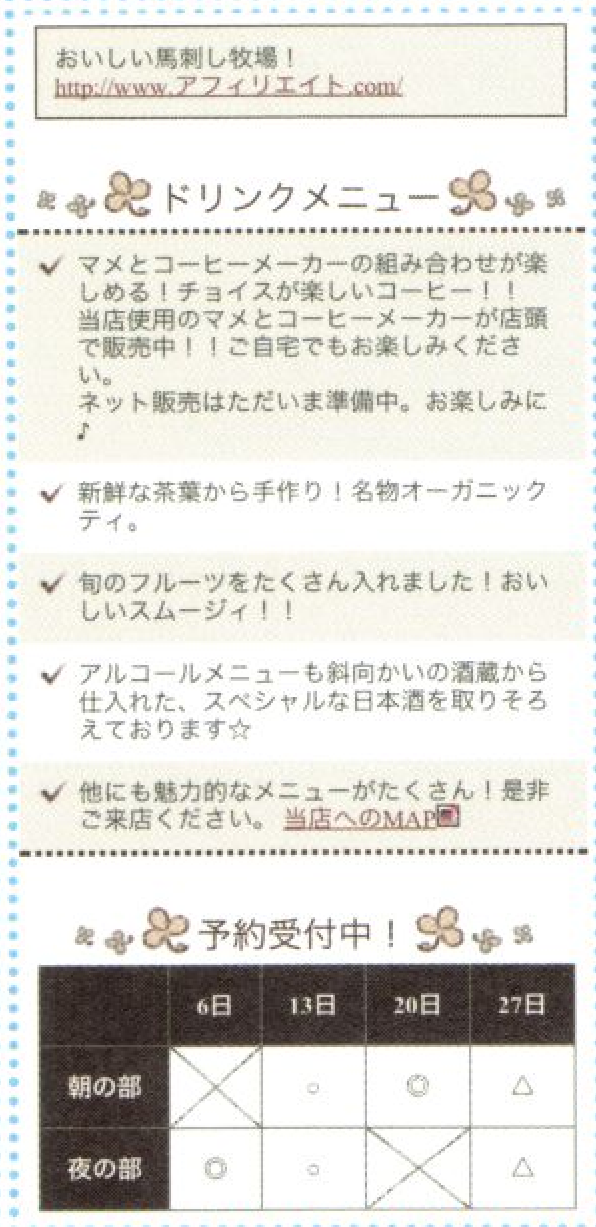
```
ul.tabMenu02 li{
float:left;
height:20px;
margin:2px 10px 0 0; padding:15px 20px 10px;
background:#ECD2D5;
border-top:#7D2A17 2px solid;border-right:#7D2A17 2px solid;border-bottom:#CCC 1px
solid;border-left:#7D2A17 2px solid;
-moz-border-radius:10px 10px 1px 1px;
-webkit-border-radius:10px 10px 1px 1px;
border-radius:10px 10px 1px 1px;
}
ul.tabMenu02 li.menuActive{
height:23px;
margin-top:0;
border-bottom:0 none;
-moz-border-radius:8px 8px 0 0;
-webkit-border-radius:8px 8px 0 0;
background:-moz-linear-gradient(#FAFAFA,#EEE);
background:-webkit-gradient(linear,left top,left bottom,from(#FAFAFA),to(#EEE));
background:linear-gradient(#FAFAFA,#EEE);
}
```

まとめ [CONCLUSION]

角丸はCSS3で利用可能になったプロパティの中でも、利用回数が多くなると思われるもののひとつです。ブラウザごとの指定が必要ですので、指定方法について把握しましょう。
また、グラデーションも利用される機会が多くなると思われますが、多用しすぎると描画が遅くなるブラウザも存在します。画像と比べてデータ量が少なくなるのは魅力ですが、表示が遅くなってしまわないように、ポイントを絞って利用しましょう。

07 属性セレクタを活用したスマートフォンサイト





特にスマートフォンサイトで活用する場面の多い属性セレクタを使ったテクニックを紹介します。idやclassは使わず、属性セレクタでファイル名を利用して画像の位置をスタイリングしたり、擬似要素と組み合わせてリンクにアイコンを表示したりする方法を解説していきます。

制作・文／佐藤とも子・サチコ（螺旋デザイン）

制作のポイント

- 属性セレクタでファイル名を利用して画像を配置
- :after 擬似要素でアイコンを表示
- :nth-child 擬似クラスでリストをスタイリング

対応ブラウザ

iOS Safari　5 以上

Android　1.6 以上

※Androidは一部の処理に非対応。

TECHNIC 1

idやclassを利用せずに画像を配置

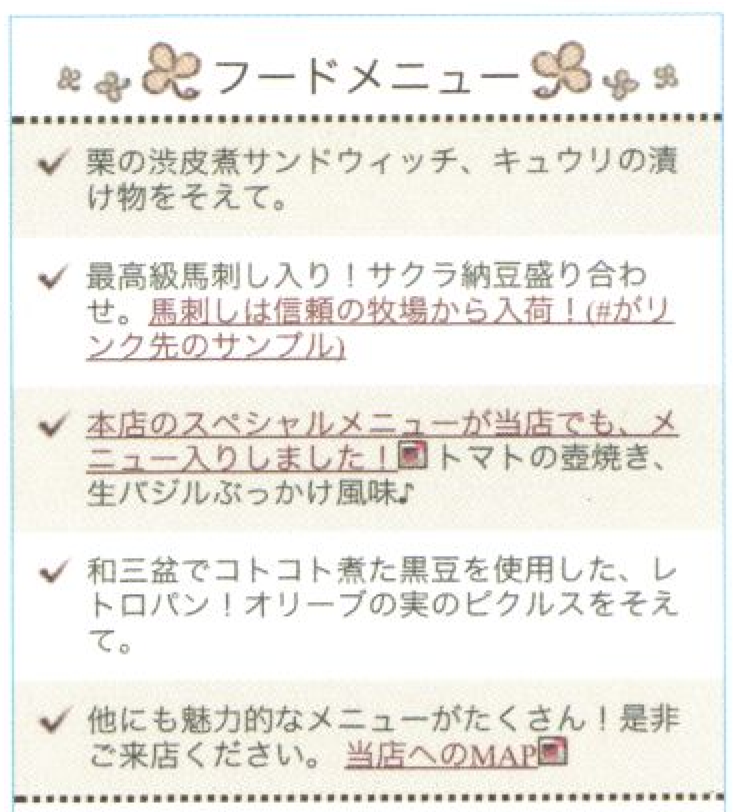
この3枚の画像は、CSS3の属性の値をセレクタにできる機能を利用して、idやclassを使わずに各画像の位置をスタイリングしています。例では画像のファイル名を利用しています。



TECHNIC 2

奇数の要素指定でリストを表現

偶数・奇数でスタイルを変更するデザインも、E:nth-child(n)を利用して特にclassを追加しなくてもCSSだけで作れます。



画像のファイル名を利用してスタイリングする

01

3枚の画像の配置に利用しているセレクタを紹介します 01-1。

CSS3で新しく策定されたセレクタに、E[foo^="homu"]というものがあります。これは、「Eの要素のfoo属性の値が"homu"から始まるもの」で指定できるのです。

具体的な例としては、画像のファイル名を利用してスタイリングする場合、img[src^="画像へのパス/ファイル名"]で指定できます 01-2。個別にidやclassを振らなくてもいいので、idやclass名を考える必要がないだけでも、だいぶ楽になります 01-3。

作例のCSSでは、ファイル名までをセレクタで指定して、拡張子は含んでいません。これは、作業中に画像フォーマットを変更しても、CSSを修正する必要がないようにしているためです。

Eと[foo^="homu"]の間には、半角スペースを入れずに詰めておきましょう。

01-1 セレクタを使用して画像を配置



01-2 CSS

```
img[src^="img/photo1"] {
position:absolute;
top: 15px;
left: 50%;
margin-left:-70px;
z-index: 10;
}
```

01-3 HTML

```
<div id="photos">
<img src="img/photo1.png" width="135" height="153" alt="和三盆でコトコト煮た黒豆を使用した、レトロパン！オリーブの実のピクルスをそえて" >
<img src="img/photo2.png" width="124" height="141" alt="栗の渋皮煮サンドウィッチ、キュウリの漬け物をそえて" >
<img src="img/photo3.png" width="124" height="141" alt="トマトの壺焼き、生バジルぶっかけ風味" ></div>
```

TIPS

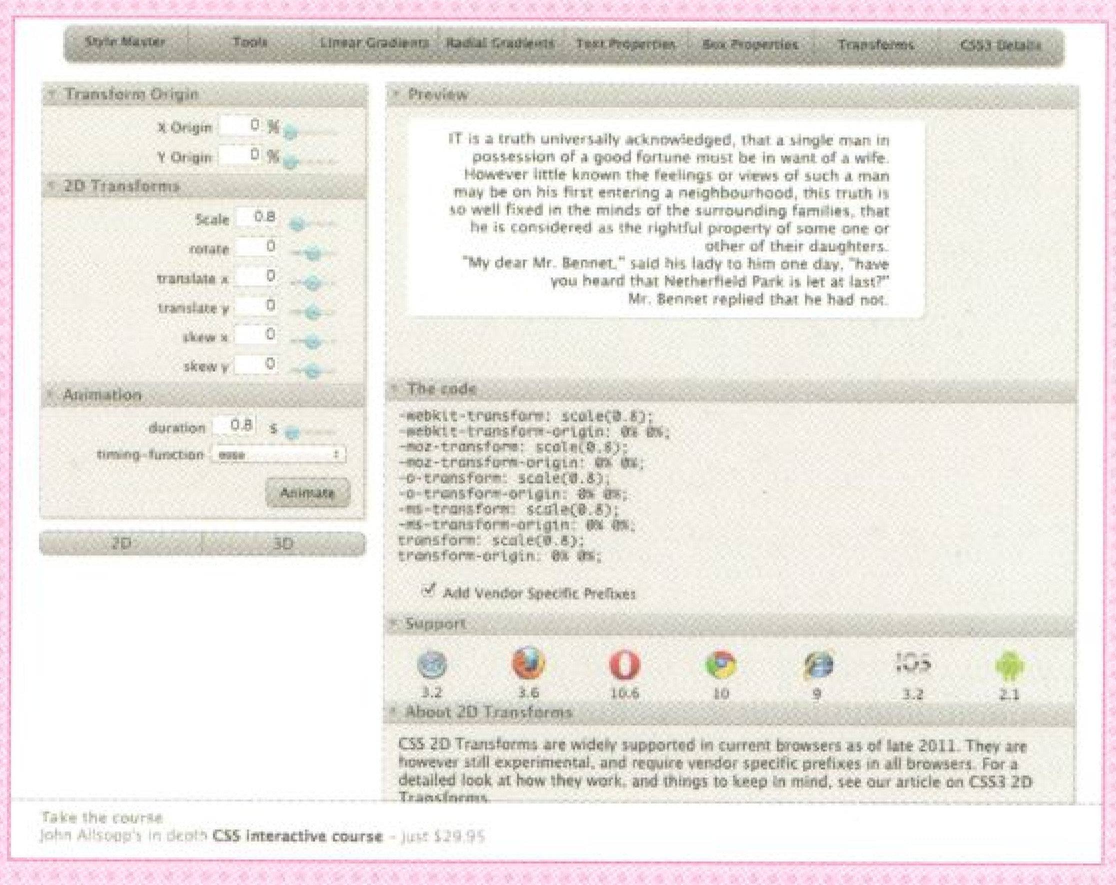
今回のサンプルでは、画像を回転させるプロパティ(-webkit-transform)を使用しています。

transformは設定項目が多いため、scaleやrotate複数の値を設定した場合の結果が予想しにくいことがありますが、01のWebサイトからCSS Transformツールを利用すると、目視しながら値を設定できます02。

01 Style MasterのCSS Transformツール

http://westciv.com/tools/transforms/

02 transformの値を目視しながら設定できる

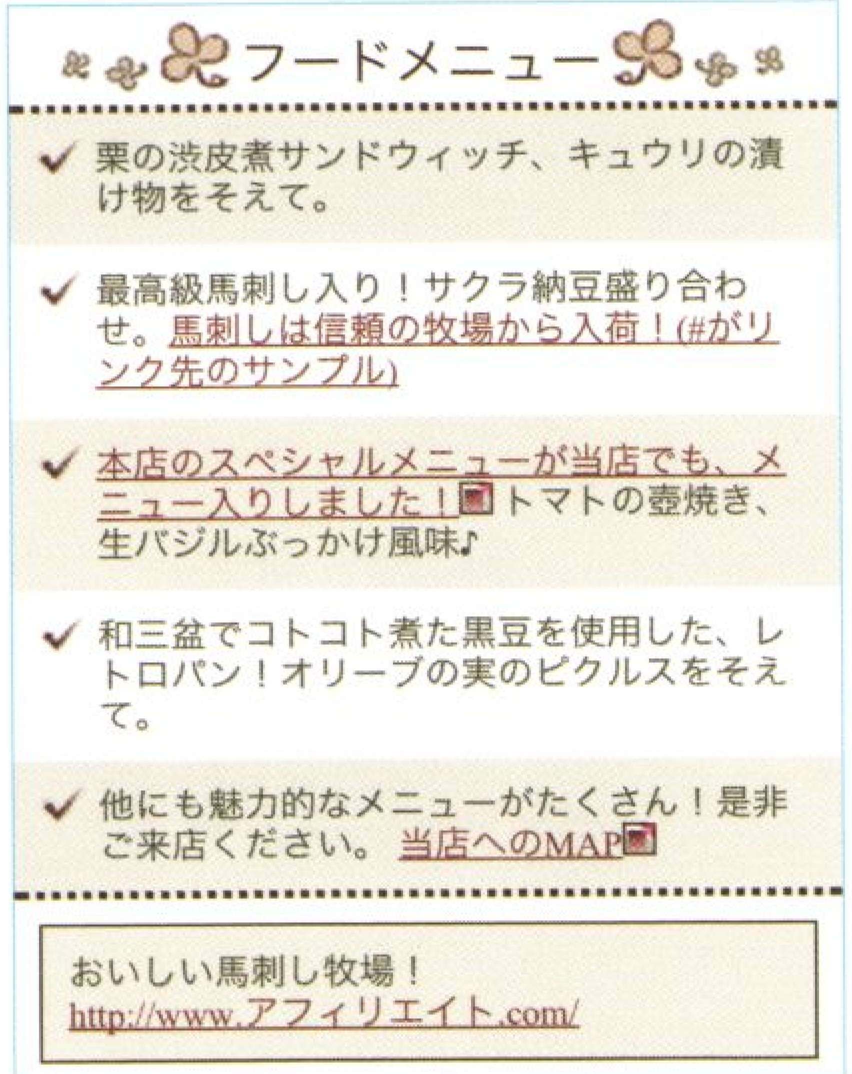


外部リンクの後ろにアイコンを表示させる

02

ページ中程の外部リンクを示すアイコンも、簡単に付けることができます 02-1。
ここでは先頭からのファイル名の指定でスタイルを付けましたが、E[foo*="homu"]を使うと、「位置は関係なく、"homu"を含む文字列を持った値」で指定することができます。
例えば、「"http://"の値を持つsrc属性のa要素」とすれば、簡単に外部リンクの指定ができます 02-2。
なお、この指定だけでいい場合は、01 で解説したE[foo^="homu"]でも支障ありません。

02-1 外部リンクを示すアイコン



02-2 CSS

```
a[href*="http://"]:after {
content:url(img/window.png);
margin: 0 3px 0 0;
}
```

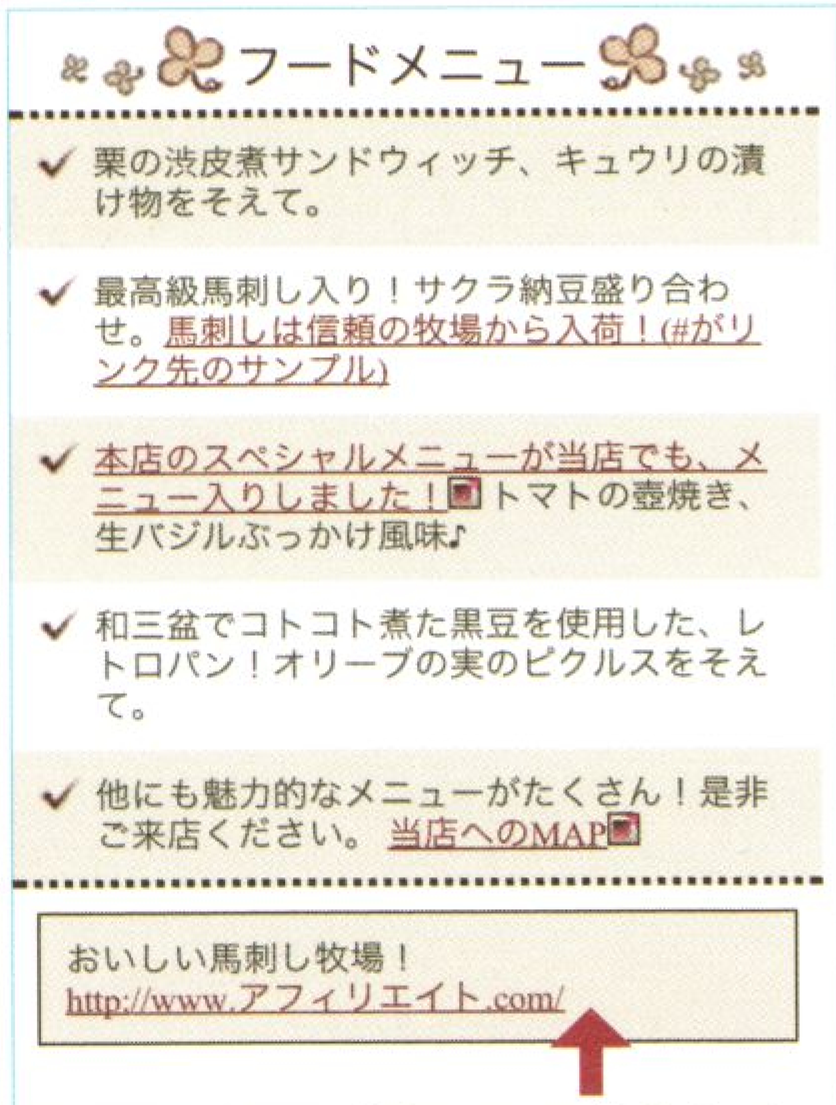
ATTENTION!

03のCSSが完成形なので、02-2 はsample.cssには記載されていません。

03

このままでもいいのですが、「アフィリエイトのリンクにはアイコンを付けたくない！」といった例外を指定したいケースもあります 03-1。
この場合は、E:not[s]での指定も併用すると、アフィリエイトのリンクを対象外にすることができます 03-2。
CMSなどを使用していて、サイト内リンクが「http://」から始まるフルパスで書かれている場合も、ドメイン名をフィルタにしてこの書式を利用するといいでしょう。

03-1 リンクにアイコンを付けない



a[href*="http://"]:after で指定するとここにもアイコンが付いてしまいます。

03-2 特定のドメイン名を対象外にする指定

```
a[href*="http://"]:not([href*="アフィリエイト"]):after {
content:url(img/window.png);
margin: 0 3px 0 0;
}
```

TIPS

ほかにも属性の値をセレクタにできる書き方があります 01。
E[foo$="homu"]を利用すると「拡張子ごとに別のアイコンを付ける」なども簡単にできますね。

01 属性セレクタの書き方

E[foo]	foo属性を持つE要素
E[foo="homu"]	foo属性の値が「homu」であるE要素
E[foo^="homu"]	foo属性の値が「homu」で始まるE要素
E[foo$="homu"]	foo属性の値が「homu」で終わるE要素
E[foo*="homu"]	foo属性の値が「homu」を部分的に含むE要素



兄弟関係の要素でマージンを調整する

04

E ~ Fで要素を指定すると、同じ親の兄弟関係の弟になる要素にスタイル付けをすることができます04-1。
CSS2では、E + Fといった指定方法も可能で、こちらも同じ親の兄弟関係の弟になる要素にスタイル付けをすることができます。2つの違いは、E ~ Fの方は親が同じであれば、間に何が入ってもスタイルが適用されます。
作例ではsectionで兄弟関係を作っていますが、間にasideを挟み変則的にしています04-2。
作例では、一番上のsectionと、その後のsectionではマージンの値が違います04-3。
E + Fは、連続していないと適用されません。

04-1 E ~ Fで兄弟関係を指定する



04-2 HTML

```
<section>
— 省略 —
</section>
<aside id="uma">
おいしい馬刺し牧場！<br />
<a href="http://www.アフィリエイト
.com/">http://www.アフィリエイト.com/</a>
</aside>
<section>
— 省略 —
</section>
<section>
— 省略 —
</section>
```

04-3 CSS

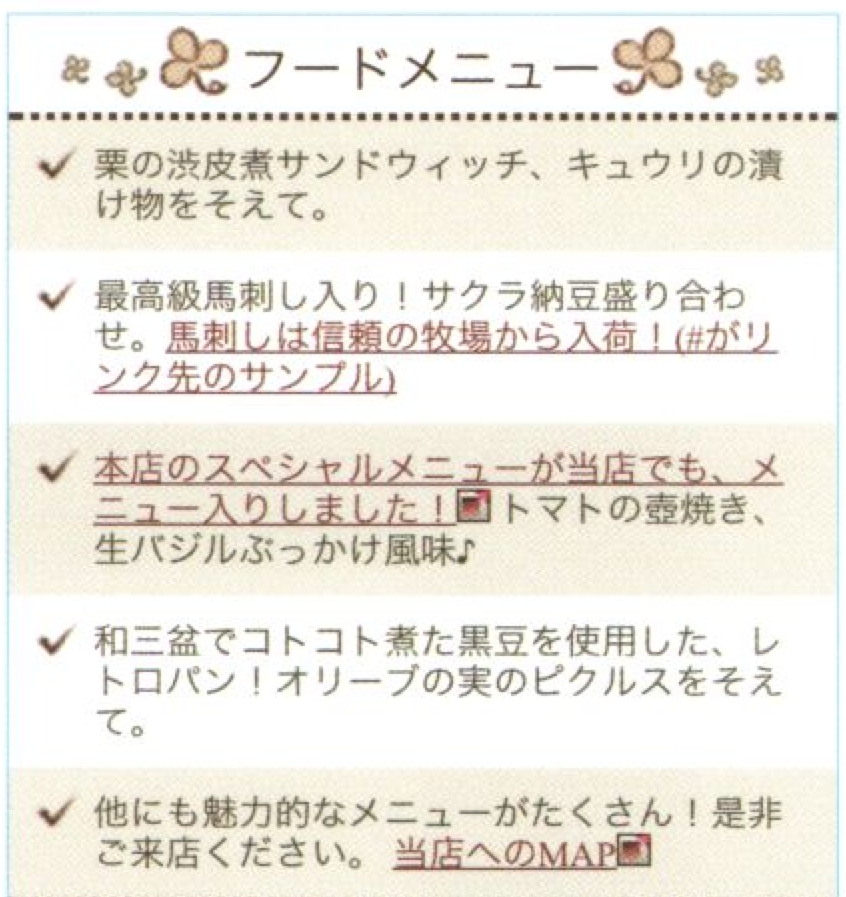
```
section {
margin: 0;
}
section ~ section {
margin-top: 30px;
}
```

奇数の要素指定を利用して“しましま”のリストを作る

05

背景色を交互に変えるしましまのリストも、簡単にできます05-1。
CSS3で新しく策定されたセレクタに、E:nth-child(n)というものがあります。これは、「親が同じEの要素のn番目」という形式で指定できるのです。
nには1や2などの決め打ち数字も入れることができますが、odd（奇数）even（偶数）といった値も入れることができます05-2。
これを利用して、しましまのリストを作ることができます。

05-1 CSSだけで作ったしましまのリスト



05-2 リストの背景色を交互に変える設定

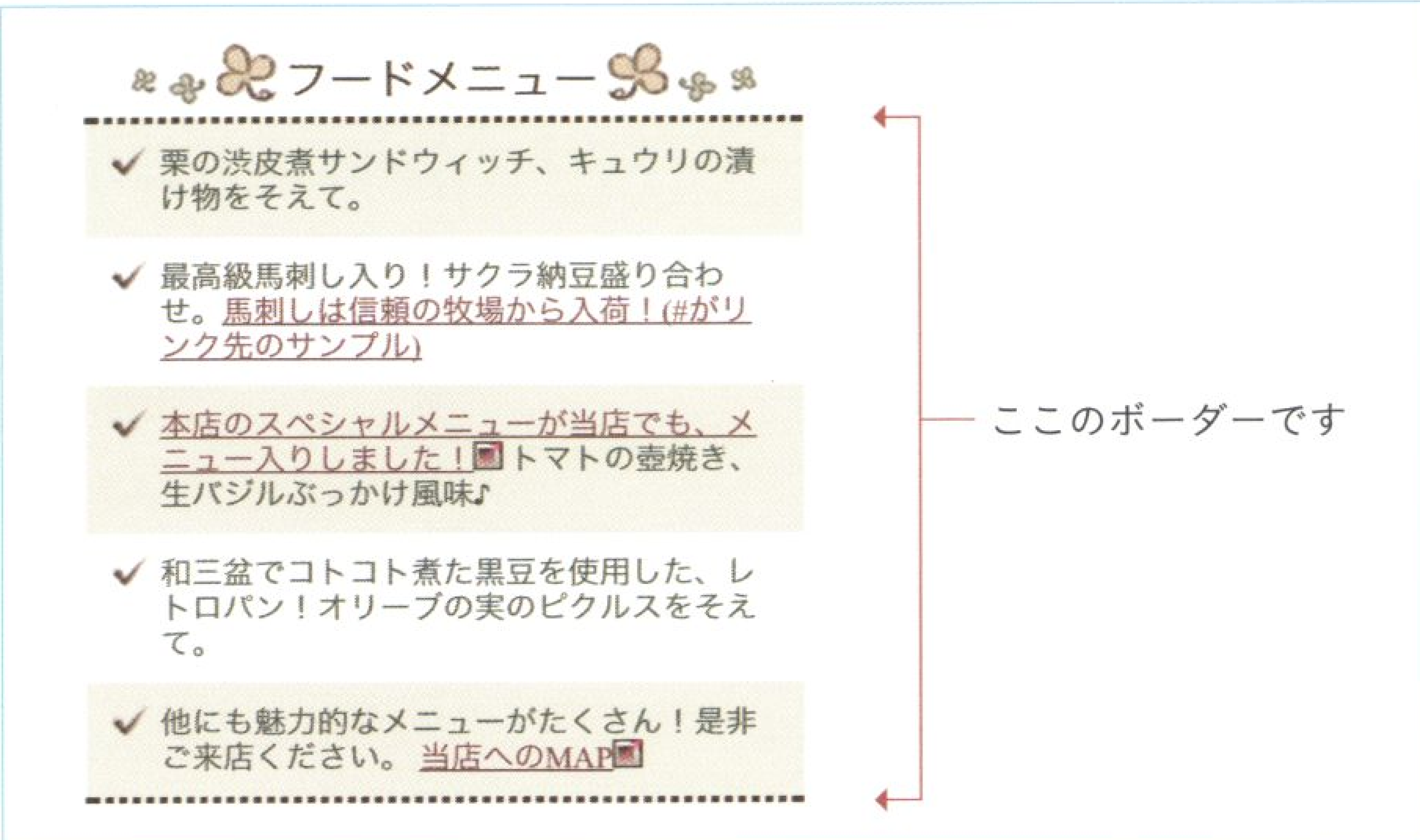
```
li:nth-child(odd) {
background-color: #EFECE7;
}
```

リストの最初と最後の要素にスタイリングする

06

次は、リストの上下にボーダーを設定します 06-1。先ほどのE:nth-child(n)を使えばできそうな気がしますが、数字を入れなくても、E:first-childとE:last-childで指定できます 06-2。最初の値を指定するE:first-childの方はCSS2からありましたが、最後の値が変動する場合に、CSS3で新しく追加されたE:last-childは便利です。

06-1 上下のボーダーを設定する



06-2 上下のボーダーを設定するCSS

```
li:first-child {
border-top: 3px #400000 dotted;
}
li:last-child {
border-bottom: 3px #400000 dotted;
}
```

空のセルに×印の画像を表示する

07

空のtdに×印を付けてみます 07-1。内容が空の要素をスタイリングできるセレクタ、E:emptyを利用して、空のセルにスタイルを付けることができます 07-2。
空の要素は通常作ることはありませんが、テーブルのセルの場合はしばしば利用することがあります。td:emptyで指定すると、空のtd要素のスタイリングを行うことができます。
ここでは同じ空要素でもthには設定していませんので、×印は付きません。

07-1 td:emptyで空のセルに×印を付ける



	6日	13日	20日	27日
朝の部	×	○	◎	△
夜の部	◎	○	×	△

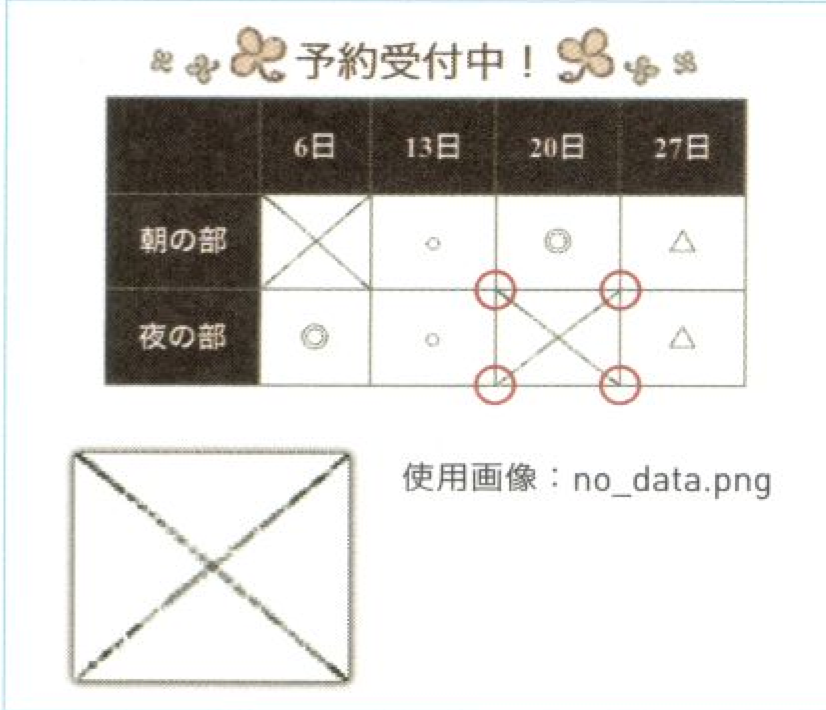
07-2 空のtdにはclassや画像は入っていない

```
<table>
<tr><th></th><th>6日</th><th>13日</th><th>20日</th><th>27日</th></tr>
<tr><th>朝の部</th><td></td><td>○</td><td>◎</td><td>△</td></tr>
<tr><th>夜の部</th><td>◎</td><td>○</td><td></td><td>△</td></tr>
</table>
```

08

空のセルに表示させた×画像は、セルの角にぴったり合うように設定しています 08-1。
これは、background-sizeというプロパティを利用しています 08-2。ここで使用した「background-size: 100% 100%」は、縦・横に100%になるように指定しています。ですので、セルのサイズのバランスが変わっても、四隅にフィットした画像が表示されます。
ほかにもさまざまな値で指定できますので、試してみてください（ただ、残念なことに、Android 1.6 Webブラウザでは効果がありません）。

08-1 ×画像をセルのサイズに合わせて表示



08-2 CSS

```
td:empty {
background: url(img/no_data.png) no-repeat center;
background-size: 100% 100%;
}
```

*COLUMN

スマートフォンサイトをチェックするには

スマートフォンサイトのチェックには各デバイスのシミュレータを使います。

iPhoneの場合

Xcodeという開発パッケージの一部である「iPhoneシミュレータ」01 を使用します。残念ながらMac（OS X Lion）のみの対応になります。Lion以前のバージョンだと、チェックできるiOSのバージョンも古くなってしまいます。

Androidの場合

Windows／Mac OS X用の「Android SDK」を使用します 02（Macの場合は「Eclipse IDE for Java Developers」03 のインストールも必要です）。インストールの途中でJDK（Javaの開発環境）の追加インストールを促すダイアログが出た場合は、そちらもインストールしてください。
Androidのほうはセッティングに1日〜半日ほどかかりますので、スケジュールに余裕をもって対応してください。

01 iPhoneシミュレータ（App Store）

http://itunes.apple.com/jp/app/xcode/id448457090?mt=12

02 Android SDK

http://developer.android.com/sdk/index.html

03 Eclipse IDE for Java Developers

http://www.eclipse.org/downloads/

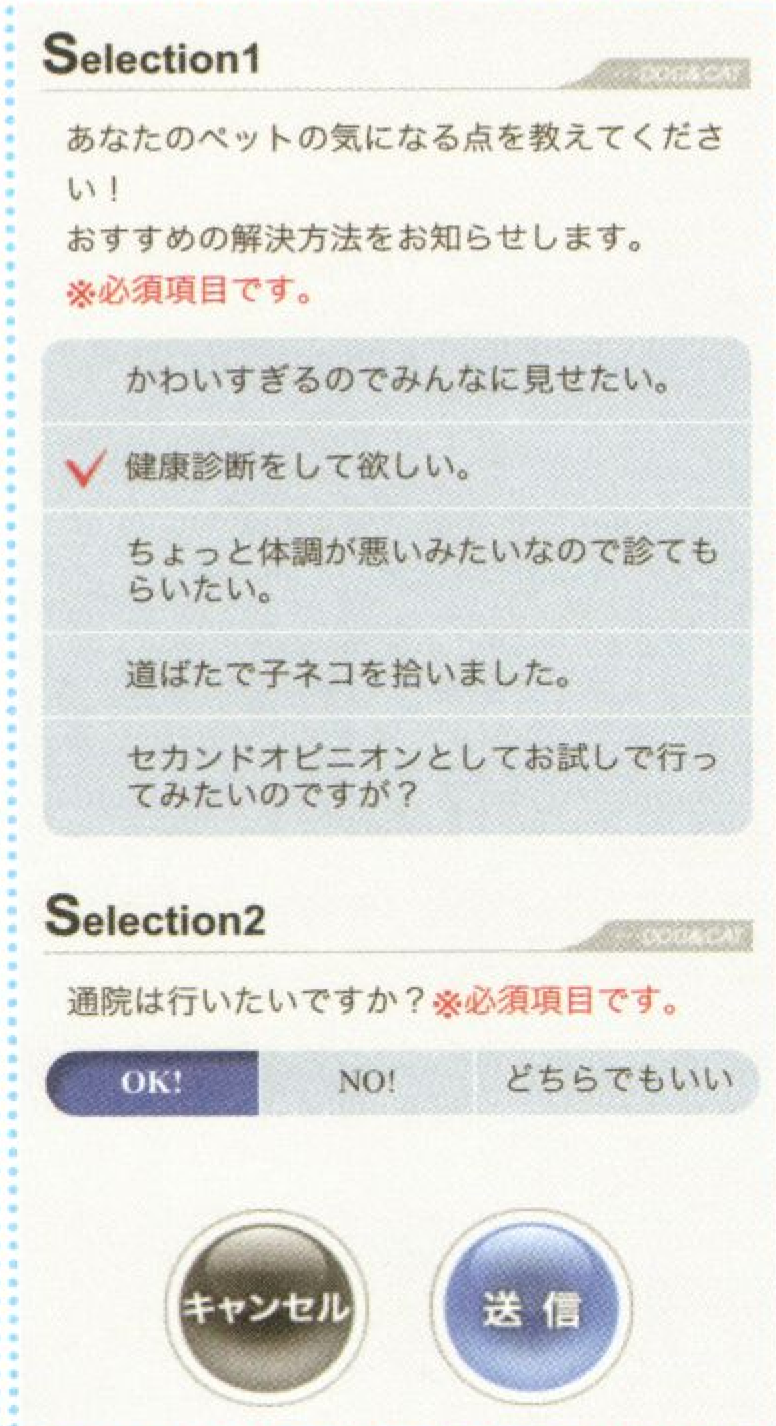
まとめ [CONCLUSION]

CSS3で新しく作られたセレクタは非常に便利ですので、積極的に利用していきましょう。ソースも単純になり、管理がしやすくなります。

CSS3のプロパティはAndroidの旧バージョンが対応していないことが多いので、利用する前に簡単に組んで確認してみてください。

08 操作性を考慮した スマートフォン向けのパーツ





スマートフォンのサイトデザインでよく見かけるUIのコーディングを、CSS3をふんだんに使用して、シンプルなソースコードで実装しました。タッチのしやすさにも配慮した設定になっています。

制作・文／佐藤とも子・サチコ（螺旋デザイン）

制作のポイント

- ■リンク範囲を広く設定
- ■擬似要素を利用してリスト部分にアイコンを表示
- ■ラジオボタンのON ／ OFFをlabel要素で表現

対応ブラウザ

iOS Safari	5 以上
Android	1.6 以上

※Androidは一部の処理に非対応。

TECHNIC 1

よく見るインターフェイスをCSSで実現

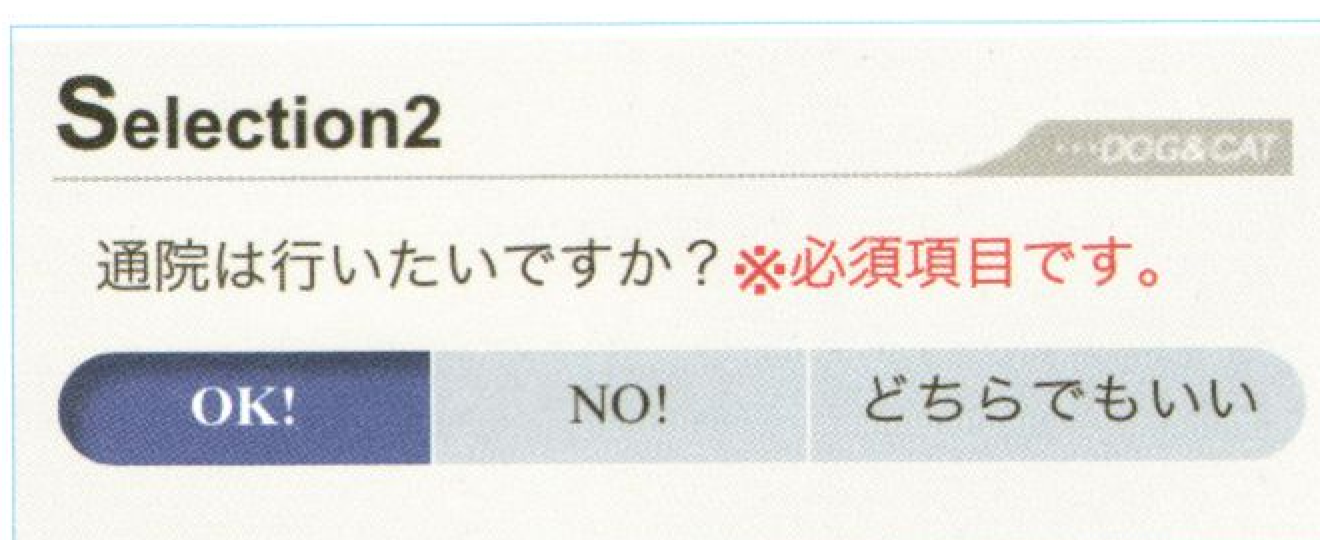
CSS3の擬似要素を駆使することで、右のようなデザインもHTML要素を飾りだけのために増やすことなく、シンプルなコードで表現できます。



TECHNIC 2

label要素にonclick属性を入れる

iOSではlabelを押しても反応しません。ここでは、label要素にonclick属性を入れることで対応しています。



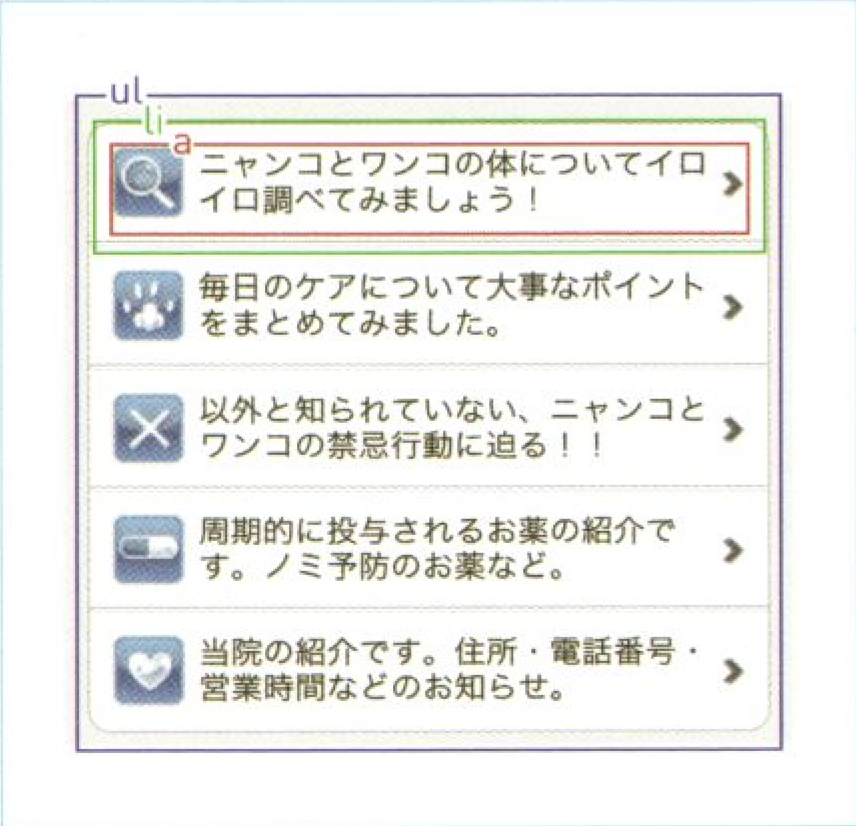


リンク範囲の広い目次のデザイン

01

スマートフォンサイトでよく見かけるタイプの、目次デザインのコーディング手法を解説します。
01-1のような構成で組みます。

01-1 目次の構成



02

まずは、ベースのulに角丸を指定します 02-1。
行全体をリンク範囲にしたいので、aタグをブロックにします 02-2。
スマートフォンではリンク範囲が大きいほど操作性が上がるので、デザイン的に不自然ではない場合はできるだけ大きくしていきましょう。

02-1 角丸の指定

```
ul#theme1 {
background-color:#FFF;
border: 1px solid #B7BCBE;
border-radius: 10px;
font-size: 14px;
line-height: 1.2;
}
```

02-2 リンク範囲を設定

```
#theme1 li a {
display:block;
padding: 10px 20px 10px 48px;
text-decoration: none;
color: #000;
position:relative;
}
```

03

aタグの:afterを利用して、矢印のアイコンをつけます 03-1。
各アイコンの位置調整は、親のaタグから絶対配置＋マージン調整で縦位置に中央になるようにしています 03-2。

03-1 矢印のアイコンをつける



03-2 :afterの設定

```
#theme1 li a:after {
content: url(img/icon_next.png);/*アイコン画像*/
position: absolute;
top: 50%;
right: 0;
margin-top:-6px;
margin-right: 10px;
}
```

TIPS

セレクタの:beforeと:after、プロパティのcontentはどちらもCSS2ですが、IE6・7で対応していないため、あまり使ったことがない方も多いのではないでしょうか。非常に便利なので、スマートフォン専用サイトなどIE6・7が対応ブラウザに含まれていない場合は是非使ってみてください。

04

a タグの :before を利用して、青いアイコンの位置指定をします 04-1。
青いアイコンはグラデーションと角丸プロパティでスタイリングしたいところですが、Android のブラウザが対応していないバージョンが多いので今回は見送りました。
青いアイコンの画像の個別指定は、:nth-child() を利用して設定しています 04-2。上から 1 〜 5 で番号を振って指定しています 04-3。class・id などを追加せずにスタイリングできるので、便利です 04-4。
以上で完成です。

04-1 :before の設定

```
#theme1 li a:before {
content: "";
position: absolute;
top: 50%;
left: 0;
margin-top:-16px;
margin-left: 10px;
width: 32px;
height: 32px;
}
```

04-2 アイコン画像の設定

```
#theme1 li:nth-child(1) a:before {
content: url(img/icon_megane.png);
}
```

04-3 1 〜 5 の番号で画像指定された



04-4 目次の HTML

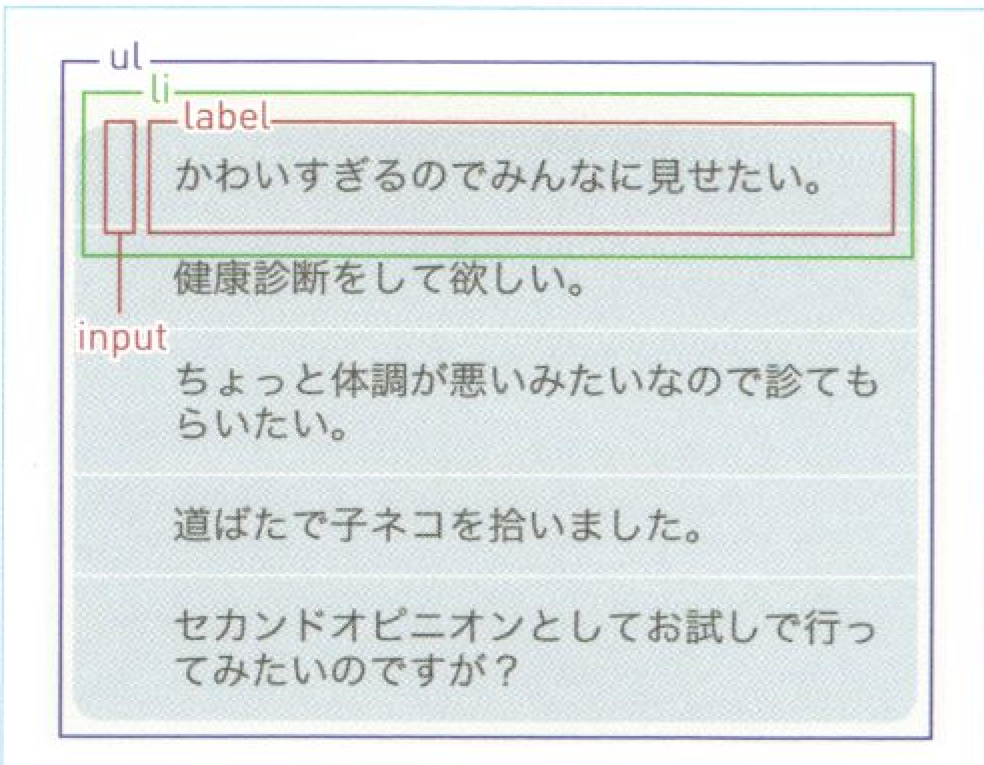
```
<ul id="theme1">
<li><a href="#">ニャンコとワンコの体についてイロイロ調べてみましょう！</a></li>
<li><a href="#">毎日のケアについて大事なポイントをまとめてみました。</a></li>
<li><a href="#">以外と知られていない、ニャンコとワンコの禁忌行動に迫る！！</a></li>
<li><a href="#">周期的に投与されるお薬の紹介です。ノミ予防のお薬など。</a></li>
<li><a href="#">当院の紹介です。住所・電話番号・営業時間などのお知らせ。</a></li>
</ul>
```

タッチしやすいラジオボタンのデザイン

05

押しやすいラジオボタンのベースを作成します。05-1 のような構成で組みます。
基本的な枠組みは先ほどの目次デザインと同様で、a タグと label タグを入れ替えたような組み方になっています。

05-1 ラジオボタンを含むページの構成





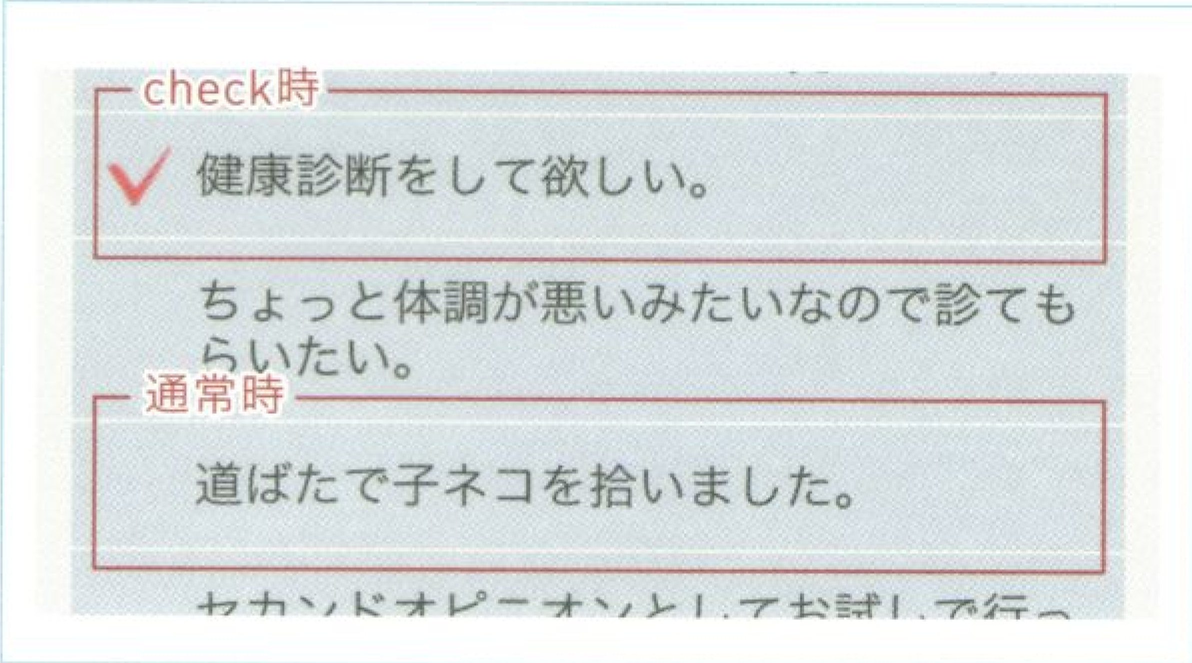
06

ラジオボタンがチェックされたら、labelのスタイルを変更するように記述しています06-1。

これは、CSS3のラジオボタンのチェックされた状態を表すセレクタの「E:checked」と、直後の要素を指定する「E + F」（これはCSS2です）を組み合わせて表現しています06-2。

ラジオボタン自体はdisplay: noneで表示を消しています06-3。

06-1 チェックマークの表示



06-2 ラジオボタンのチェック時に表示を変更する設定

```
#theme3 li input[type="radio"]:checked + label {
background: #1466B5;
color: #FFF;
font-weight: bold;
box-shadow: 5px 5px 3px rgba(0,0,0,0.3) inset;
}
```

06-3 ラジオボタンを非表示にする設定

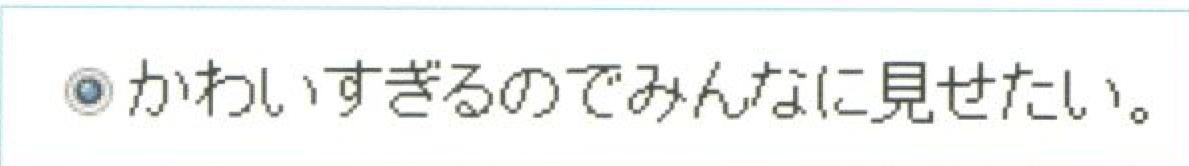
```
#theme2 li input[type="radio"], #theme3 li input[type="radio"] {
display: none;
}
```

07

PC用のサイト制作ではよく使う、labelでマーキングされた箇所をクリックすると対応するラジオボタンが反応する状態にします07-1。

ラジオボタン側のidとlabelのforの値を揃えればいいのですが07-2、このままではiOSのSafariだけ反応しません。PC用のブラウザ（Safariも含む）とAndroidではラジオボタンは反応するため、まぎらわしいので注意しましょう。

07-1 指定した範囲をクリックするとラジオボタンが反応する



07-2 sample.html

```
<input type="radio" name="s1" id="s1_1" /><label for="s1_1">かわいすぎるのでみんなに見せたい。</label>
```

08

これはlabelに空のonclickをつけるとiOSのSafariでも反応するようになります08-1。

プレーンな状態のラジオボタン＋labelでも起こる現象なので、iOS用のフォームを作成する場合は、labelに空のonclickを付けておくことで対応するといいでしょう。

08-1 空のonclickを追加

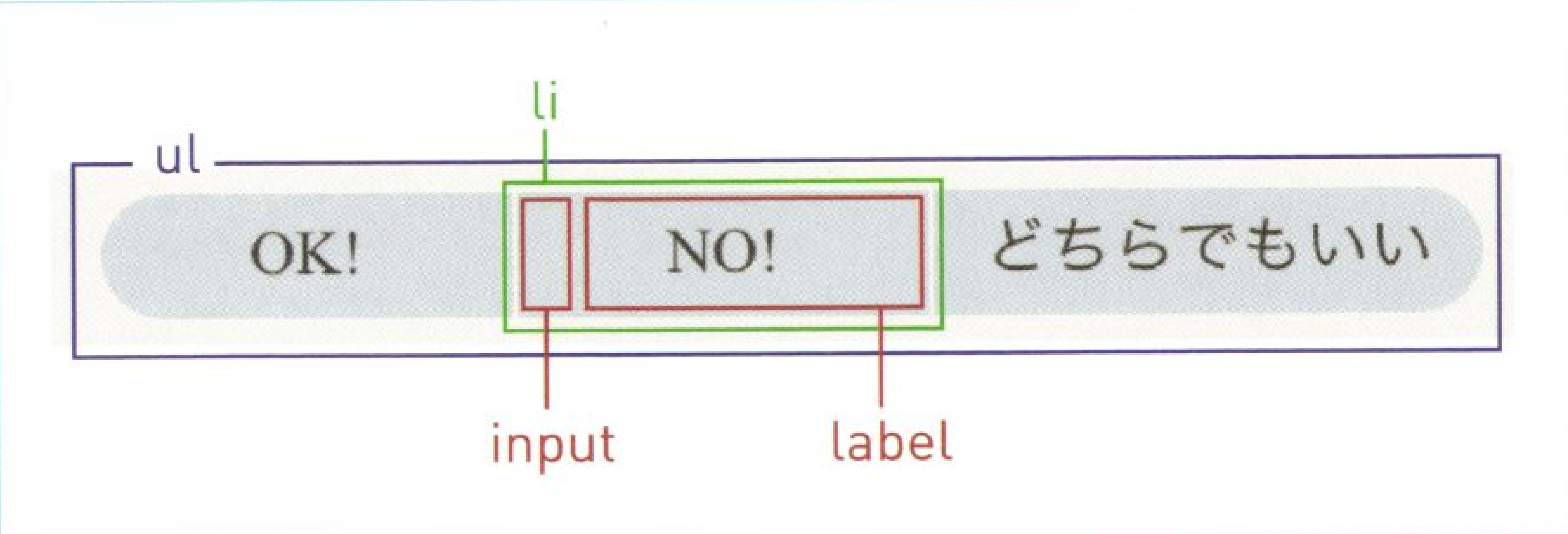
```
<input type="radio" name="s1" id="s1_1" /><label for="s1_1" onclick="">かわいすぎるのでみんなに見せたい。</label>
```

ラジオボタンを横に並べてスマートフォンらしさを出す

09

押しやすいラジオボタンの変形版です。09-1のような構成で組みます。先ほどのラジオボタンのデザインを、横並びにした状態です。
このデザインも非常にスマートフォンらしさが出ているので、選択肢の文言が短いとき適しています。

09-1 横並びにしたラジオボタン



10

08の縦並びのラジオボタンから、横に配置したいli要素の親のulでbox表示をするように指定しています10-1。「display: -webkit-box」でマーキングされた要素の子要素は自動的に横並びになり、floatの利用と違って高さも揃うので非常に便利です。
この場合はulにbox指定しているので、その子要素のliが横並びになります10-2。

10-1 ulをbox指定

```
ul#theme3 {
  margin: 0;
  display: -webkit-box;
}
```

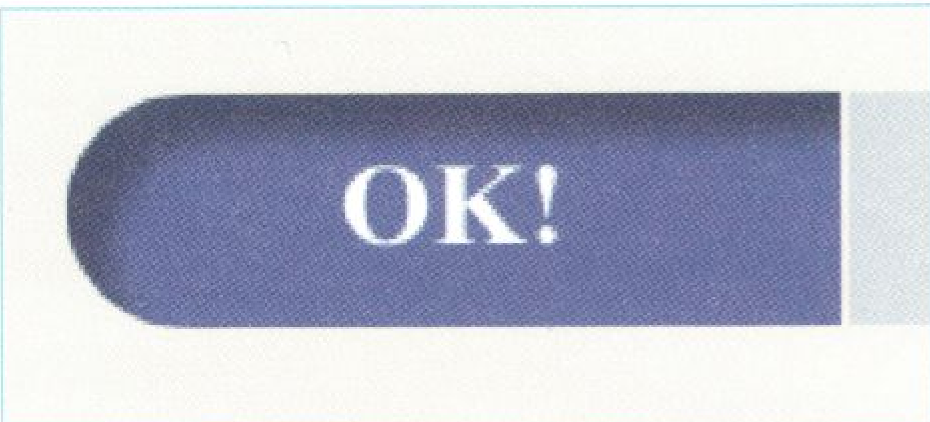
10-2 HTML

```
<ul id="theme3">
<li><input type="radio" name="s2" id="s2_1" /><label for="s2_1" onclick="">OK!</label></li>
<li><input type="radio" name="s2" id="s2_2" /><label for="s2_2" onclick="">NO!</label></li>
<li><input type="radio" name="s2" id="s2_3" /><label for="s2_3" onclick="">どちらでもいい</label></li>
</ul>
```

11

チェックされたlabelは要素の内側に影が入るようにし、ボタンが押された感を出すようにしています11-1。
これはCSS3のプロパティである「box-shadow」を利用しています。
ボックス要素の外側だけではなく、内側にも入れることができます11-2。

11-1 ラジオボタンが押された状態



11-2 box-shadowの設定

```
#theme3 li input[type="radio"]:checked + label {
background: #1466B5;
color: #FFF;
font-weight: bold;
box-shadow: 5px 5px 3px rgba(0,0,0,0.3) inset;
}
```

ATTENTION!

角丸のCSS3プロパティ（border-radius)、シャドウをつけるプロパティ（box-shadow）は、Android 1.6では対応していません。
今回のサンプルは、Android 1.6では上記の効果が表示されませんが、アクセシビリティは確保されているため、特に対策はしていません（Android 1.6では、該当箇所は直角・影なしで表示されます）。
また、グラデーションの表現もAndroid 4からの対応になりますので、気をつけて作成してください。

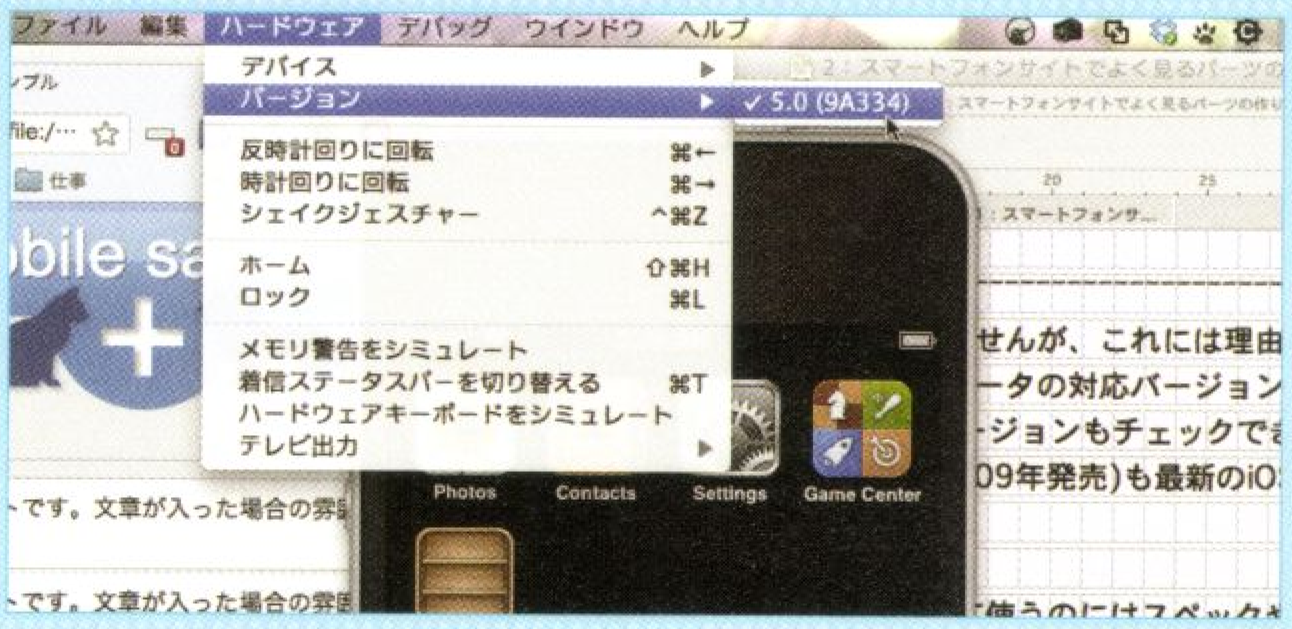


*COLUMN

iOSとAndroid OSのバージョン対応

スマートフォント専用サイトの場合、PCサイトほど古いバージョンのブラウザ対策に苦労することは少ないでしょう。
iOSは3年前の機種であるiPhone 3GS（2009年発売）でも、最新のiOS5にバージョンアップできるため、iOS3以下の環境で閲覧しているユーザーは皆無ではありませんが、少数であるといえます。
Androidは発売時のOSからアップデートできない機種があるため、ユーザーの閲覧環境には最新のAndroid OSと古いバージョンのAndroid OSが混在していることになります（Googleが新しいOSを発表しても、端末のメーカーが対応しないとアップデートできません）。
iOSは制作時の最新版に対応するように制作すればよいケースも多いですが、Android OSは制作時に最もシェアの多いバージョンの調査をしておくといいでしょう。

01 iOSシミュレータもiOS 4以下には対応していない

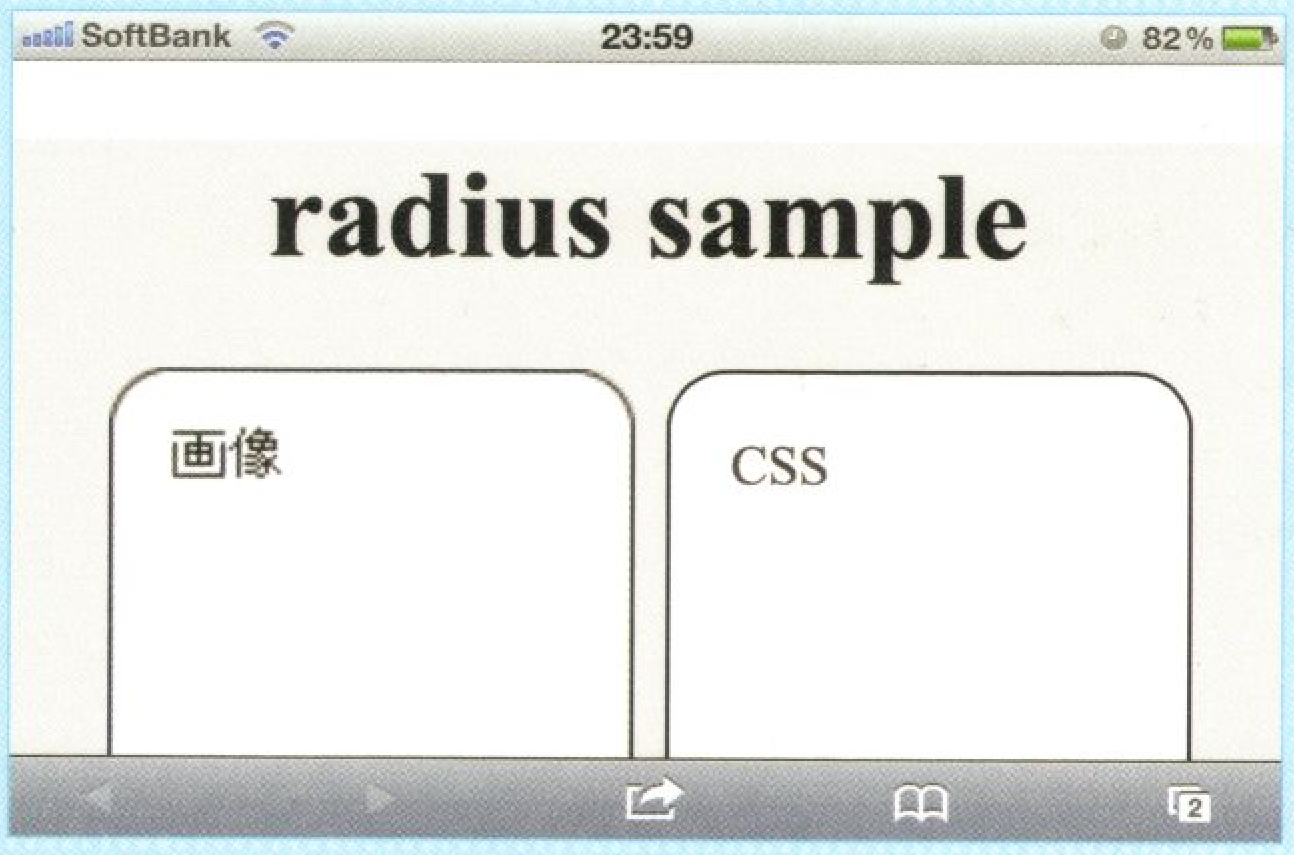


*COLUMN

試してほしいグラデーションと角丸

グラデーション（-webkit-radial-gradient）と角丸（border-radius）のプロパティは、Android OSのブラウザではまだ対応しないため、このサンプルではグラデーションを画像を使って表現しました。
今後Androidブラウザでも対応が進んだ場合や、ユーザーの閲覧環境をiPhone（Mobile Safari）に限定できる案件では、グラデーションや角丸を画像ではなくCSS3で実装してみてください。
最近のスマートフォンは画面解像度が非常に高いため、画像を利用している場合はどうしてもぼやけた印象になりがちですが、CSSを使って表現したグラデーションや角丸は、高解像度のスマートフォンでも美しく表示されます。また、拡大・縮小して表示しても画像の場合と違ってドットが目立ちません 01。

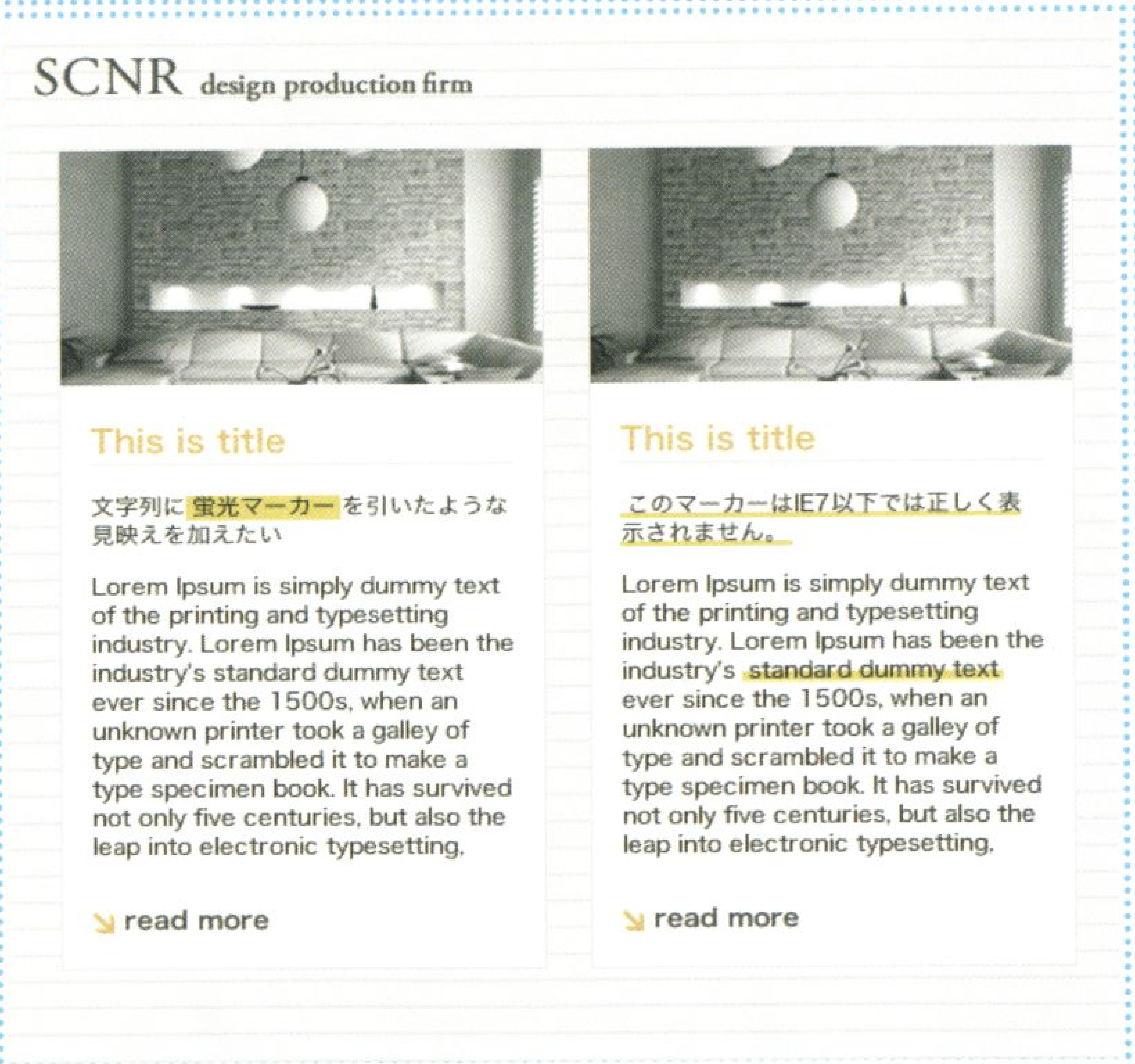
01 CSSで描画した方はドットが目立たない



まとめ [CONCLUSION]

スマートフォンサイトのデザイン・コーディングをするときは、指でタッチして操作することを頭に置きながら、ボタン系を心持ち大きめに作っていくと使いやすいサイトになります。
PC用のサイトといろいろ勝手が違い難しいところもありますが、ロールオーバーが存在しないため、その点は手間が少なくなるといえます。

09 グラデーション機能を使った蛍光マーカー風の文字装飾



文字列に蛍光ペンを引いたようなマーカー効果を施したいとき、従来は文字列の背景に色を付けたり、背景画像を使いましたが、ここではCSS3のグラデーション機能を使う方法を紹介します。マウスで選択した部分だけマーキングすることもできます。

制作・文／山本圭助
（株式会社ラナデザインアソシエイツ）

制作のポイント

- 従来のマーカー表現
- CSS3でのマーカー表現
- 選択したテキストへのマーカー表現

対応ブラウザ

IE	P.E.
Safari	5 以上
Firefox	8 以上
Chrome	15 以上

※IE7以上で表示は確保。

TECHNIC 1

CSS3のグラデーション機能を使う

CSS3のグラデーション機能を使い、色の変化をあえてさせないことで、マーキング効果を実現します。

> 文字列に蛍光マーカーを引いたような見映えを加えたい
>
> Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting,

TECHNIC 2

マウスで選択した部分にマーキングを施す

::selection擬似要素を使って、マウスで指定した部分にインタラクティブなマーキングを行います。

> このブロックのみ、テキストを選択するとマーカー色が付きます。
>
> Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting,



従来のマーカー表現

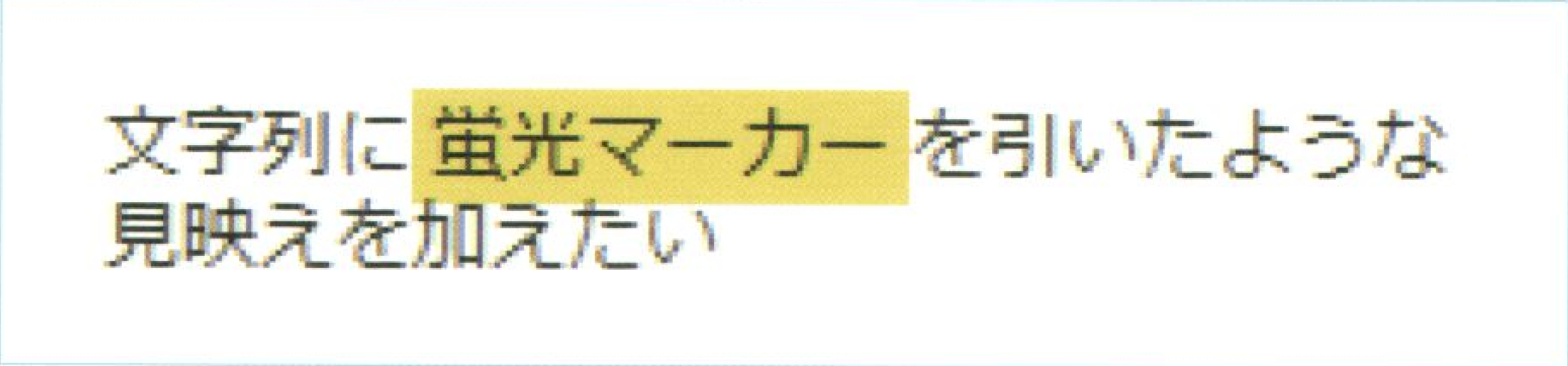
01

文字列に蛍光マーカーを引いたような見映えを加えたい場合、文字列の背景色を変更します 01-1 01-2。

01-1 従来の方法

```
// html
<p>
  文字列に<em>蛍光マーカー</em>を引いたような見映えを加えたい
</p>
// css
em {
  background-color: #fadd4b;
}
```

01-2 従来の方法によるマーカー表現



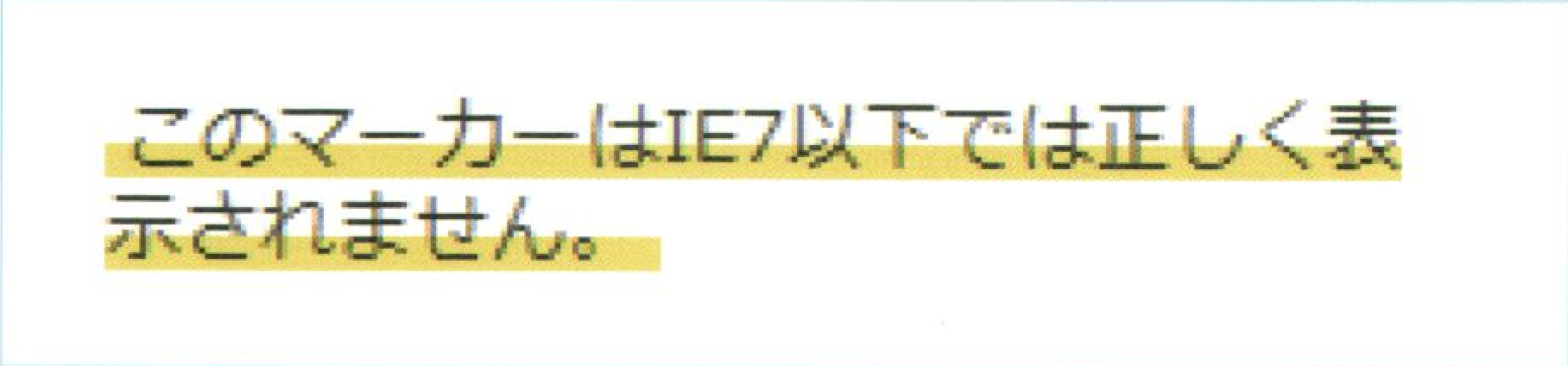
02

もうひと工夫したい場合は、背景画像を使う方法があります 02-1 02-2。ただし、この方法は行の折り返しを伴う場所では、IE7以下では正しく表示されません 02-3。

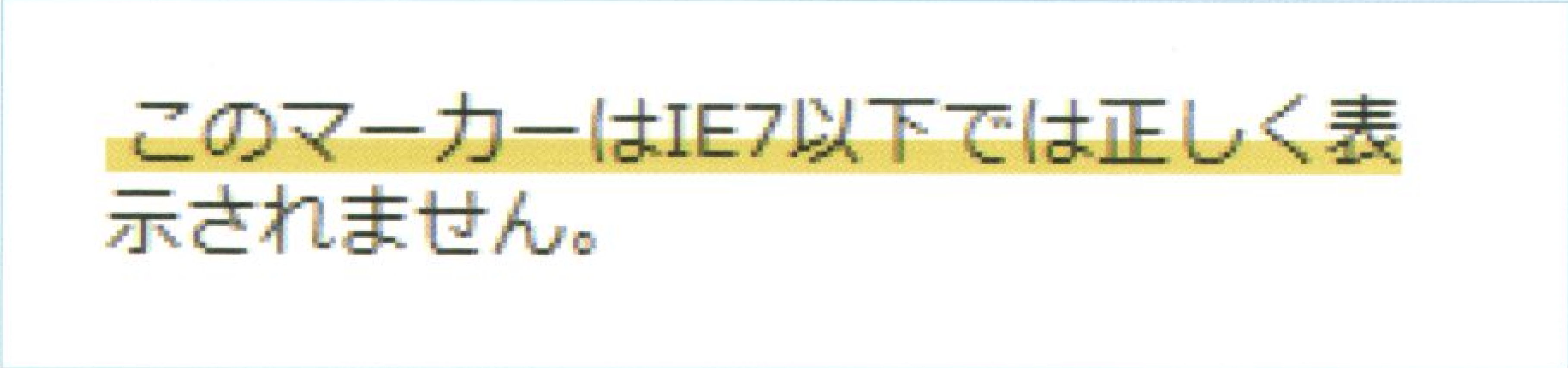
02-1 背景画像を使う方法

```
// html
<p>
  <strong>このマーカーはIE7以下では正しく表示されません。</strong>
</p>
// css
strong {
  background: url(../img/marker-bg.png) repeat-x 0 4px;
}
```

02-2 背景画像を使ったマーカー表現（Firefox、Chrome）



02-3 背景画像を使ったマーカー表現（IE7、IE6）



CSS3でのマーカー表現

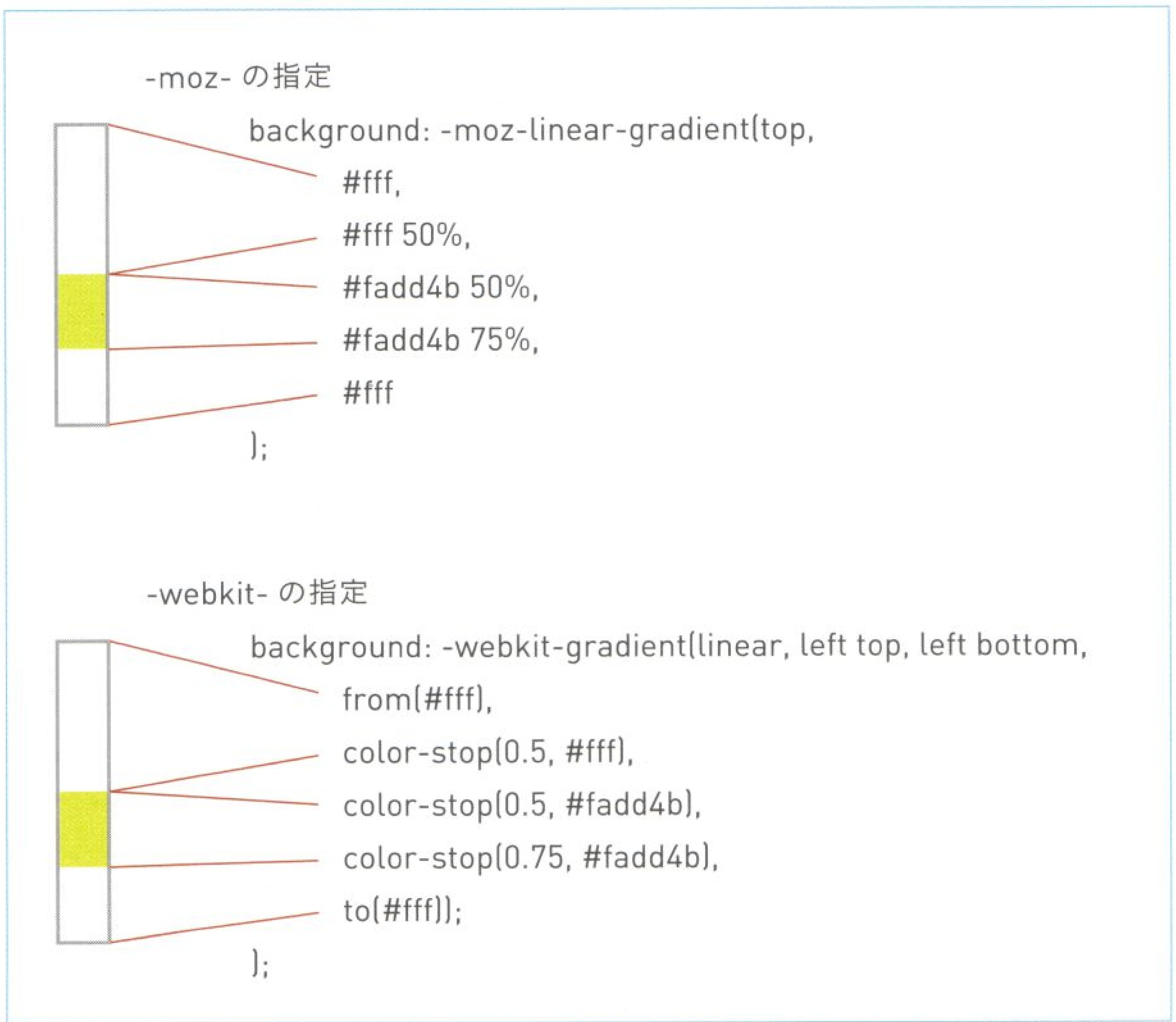
03

CSS3のグラデーション機能を使えば、画像を指定したのと同様の表示を行うことができます 03-1 。
色の境目に2色を指定することで、あえてグラデーションさせず、色を切り替えます 03-2 。

03-1 CSS3のグラデーション機能を使う方法

```
// html
<p>
  文字列に<em>蛍光マーカー</em>を引いたような見映えを加えたい
</p>
// css
em {
  // グラデーション非対応ブラウザ用
  background-color: #fadd4b;
  // Firefox用
  background: -moz-linear-gradient(top,
    #fff,
    #fff 50%,
    #fadd4b 50%,
    #fadd4b 75%,
    #fff
  );
  // Chrome, Safari用
  background: -webkit-gradient(linear, left top, left bottom,
    from(#fff),
    color-stop(0.5, #fff),
    color-stop(0.5, #fadd4b),
    color-stop(0.75, #fadd4b),
    to(#fff)
  );
}
```

03-2 ブラウザごとのグラデーション指定





MEMO

Chrome と Safari は指定の方法が2種類あります。

Firefox
-moz-linear-gradient(方向, 開始色, 中間色 位置(%), 終了色);
※中間色は省略可能。位置は%で指定する。

Chrome ／ Safari
-webkit-gradient(linear, 開始位置, 終了位置, from(開始色), color-stop(位置, 中間色), to(終了色));
※中間色は省略可能。位置は%か0〜1の小数値で指定する。

-webkit-linear-gradient(方向, 開始色, 中間色 位置(%), 終了色);
※中間色は省略可能。位置は%で指定する。

選択したテキストへの装飾

04

::selection 擬似要素を使うことで、選択した文字列の色を変更することが可能です 04-1。

04-1 ::selection 擬似要素を使う方法

```
// html
<p>
  テキストを選択するとマーカー色が付きます。
</p>
// css
::selection {
  background-color: #fadd4b;
}
::-moz-selection { // Firefox用
  background-color: #fadd4b;
}
```

ATTENTION!

2011年9月29日付けのセレクタに関するW3C勧告では、::selection 擬似要素は仕様から外されています 01。

01 W3Cの勧告

http://www.w3.org/TR/css3-selectors/#selection

まとめ [CONCLUSION]

グラデーション機能であえてグラデーションさせないことで、背景の任意の位置にラインを描くことができます。
一般的に下線を使って存在を示すことが多いa要素のリンクテキストや、打ち消し線を使用するdel要素のテキストなども、グラデーション機能を使うことで、表現力を上げることが可能になります。
いろいろな場所で応用してみてください。

目的別索引

用語索引

執筆
赤間公太郎（株式会社マジカルリミックス）／新谷剛史（株式会社セカンドファクトリー）
河原 覚（株式会社アークフィリア）／境 祐司／佐藤とも子・サチコ（螺旋デザイン）
執行大介／高橋としゆき（Graphic Arts Unit）／高村香奈子（株式会社アークフィリア）
中西祐介（株式会社ワン・トゥー・テン・デザイン）／藤本勝己（株式会社レリーフ）
山田敬美／山本圭助（株式会社ラナデザインアソシエイツ）

制作スタッフ
装丁・本文デザイン　山浦隆史（Concent, Inc.）
DTP　株式会社トップスタジオ

副編集長　後藤憲司
編集　熊谷千春、後藤孝太郎、株式会社トップスタジオ

CSS3デザインブック
仕事で絶対に使うプロのテクニック

2012年4月1日　初版第1刷発行

編者　MdN編集部
発行人　藤岡 功
発行　株式会社エムディエヌコーポレーション
〒102-0075　東京都千代田区三番町20
http://www.MdN.co.jp/
発売　株式会社インプレスコミュニケーションズ
〒102-0075　東京都千代田区三番町20
TEL：03-5275-2442　FAX：03-5275-2444（出版営業）
印刷・製本　株式会社リーブルテック

Printed in Japan

造本には万全を期しておりますが、万一、落丁・乱丁などがございましたら、送料小社負担にてお取り替えいたします。お手数ですが、カスタマーセンターまでご返送ください。

落丁・乱丁本などのご返送先
〒102-0075　東京都千代田区三番町20
株式会社エムディエヌコーポレーション カスタマーセンター
TEL：03-4334-2915

内容に関するお問い合わせ先
株式会社エムディエヌコーポレーション カスタマーセンター メール窓口
info@MdN.co.jp
本書の内容に関するご質問は、Eメールのみの受付となります。メールの件名は「CSS3デザインブック 仕事で絶対に使うプロのテクニック 質問係」、本文にはお使いのマシン環境（OS、使用ブラウザなど）をお書き添えください。電話やFAX、郵便でのご質問にはお答えできません。ご質問の内容によりましては、しばらくお時間をいただく場合がございます。また、本書の範囲を超えるご質問に関しましてはお答えいたしかねますので、あらかじめご了承ください。

MdNのWeb関連の本

ウェブレイアウトの教科書
PC・スマートフォン・タブレット時代の標準デザイン

境祐司 著
B5変型判／240ページ
ISBN978-4-8443-6241-8／定価（本体2,500円＋税）

iPhone & Android
スマートフォンサイト制作の教科書

こもりまさあき、谷拓樹、石本光司、いちがみトモロヲ 共著
B5変型判／224ページ
ISBN978-4-8443-6223-4／定価（本体2,400円＋税）

ウェブデザインのつくり方、
インターフェイスデザインの考え方。

矢野りん 著
A5判／256ページ
ISBN978-4-8443-6222-7／定価（本体2,300円＋税）

iPhone & Android
HTML5ではじめるアプリ制作の手引き

外村和仁、大村徹、徳田和規、クロカワ リュート、清水大輔 共著
B5変型判／320ページ
ISBN978-4-8443-6208-1／定価（本体2,800円＋税）

ISBN978-4-8443-6253-1
C3055 ¥2800E

9784844362531

1923055028005

定価（本体2,800円+税）
発行 エムディエヌコーポレーション
発売 インプレスコミュニケーションズ

CSS3
デザインブック
仕事で絶対に使うプロのテクニック

web creators

web creators
CSS3 デザインブック
仕事で絶対に使うプロのテクニック

MdN Corporation

DOWN LOAD